TLM 2.0 Glossary

Version 0.2

June 1st, 2007

Bugs and Suggestions

Please report bugs and suggestions about this document to

http://www.systemc.org

[Draft OSCI TLM LRM]

(Informative)

Blue = from the SystemC LRM

Glossary

This glossary contains brief, informal descriptions for a number of terms and phrases used in this standard. Where appropriate, the complete, formal definition of each term or phrase is given in the main body of the standard. Each glossary entry contains either the clause number of the definition in the main body of the standard or an indication that the term is defined in ISO/IEC 14882:2003 or IEEE Std 1666™-2005.

adaptor: ---

approximately timed: A model for which there exists a one-to-one mapping between the externally observable states of the model and the states of some corresponding detailed reference model such that the mapping preserves the sequence of state transitions but not their precise timing. The degree of timing accuracy is undefined. See *cycle approximate*.

attribute (of a transaction): Data that is part of and carried with the transaction and is implemented as a member of the transaction object. These may include attributes inherent in the bus or protocol being modeled, and attributes that are artefacts of the simulation model (a timestamp, for example).

bidirectional interface: A TLM 1.0 transaction level interface in which a pair of transaction objects, the request and the response, are passed in opposite directions, each being passed according to the rules of the unidirectional interface. For each transaction object, the transaction attributes are strictly readonly in the period between the first timing point and the end of the transaction lifetime.

blocking: Permitted to call the **wait** method. A blocking function may consume simulation time or perform a context switch, and therefore shall not be called from a method process. A blocking interface defines only blocking functions.

bridge: A module that connects together two similar or dissimilar transaction-level interfaces, each representing a memory-mapped bus or other protocol, usually at the same abstraction level. A bus bridge is a device that connects two similar or dissimilar buses together. A communication bridge is a device that connects network segments on the data link layer of a network. See *transactor*.

caller: In a function call, the sequence of statements from which the given function is called. The referent of the term may be a function, a process, or a module.

callee: In a function call, the function that is called by the caller.

channel: A class that implements one or more interfaces or an instance of such a class. A channel may be a hierarchical channel or a primitive channel or, if neither of these, it is strongly recommended that a channel at least be derived from class **sc_object**. Channels serve to encapsulate the definition of a communication mechanism or protocol. (SystemC term)

core interface: One of the [N] specific transaction level interfaces defined in this standard that are not derived from any other interface. Each core interface is an *interface proper*.

cycle accurate: A modeling style in which it is possible to predict the state of the model in any given cycle at the external boundary of the model and thus to establish a one-to-one correspondence between the states of the model and the externally observable states of a corresponding RTL model in each cycle, but which is not required to explicitly re-evaluate the state of the entire model in every cycle or to explicitly represent the state of every boundary pin or internal register. This term is only applicable to models that have a notion of cycles.

cycle approximate: A model for which there exists a one-to-one mapping between the externally observable states of the model and the states of some corresponding cycle accurate model such that the mapping preserves the sequence of state transitions but not their precise timing. The degree of timing accuracy is undefined. This term is only applicable to models that have a notion of cycles.

cycle count accurate, cycle count accurate at transaction boundaries: A modeling style in which it is possible to establish a one-to-one correspondence between the states of the model and the externally observable states of a corresponding RTL model as sampled at the timing points marking the boundaries of a transaction. A cycle count accurate model is not required to be cycle accurate in every cycle, but is required to accurately predict both the functional state and the number of cycles at certain key timing points as defined by the boundaries of the transactions through which the model communicates with other models.

declaration: A C++ language construct that introduces a name into a C++ program and specifies how the C++ compiler is to interpret that name. Not all declarations are definitions. For example, a class declaration specifies the name of the class but not the class members, while a function declaration specifies the function parameters but not the function body. (See definition.) (C++ term)

definition: The complete specification of a variable, function, type, or template. For example, a class definition specifies the class name and the class members, and a function definition specifies the function parameters and the function body. (See declaration.) (C++ term)

initiator: This term has no precise technical definition in this standard, but is used to mean a component that can initiate transactions on a memory-mapped bus or network-on-chip. An initiator is usually a master and a master an initiator, but the term *initiator* means that a component can initiate transactions, whereas the term *master* means that a component can take control of a bus.

interface: A class derived from class **sc_interface**. An interface proper is an interface, and in the object-oriented sense a channel is also an interface. However, a channel is not an interface proper. (SystemC term)

Interface Method Call (IMC): A call to an interface method. An interface method is a member function declared within an interface. The IMC paradigm provides a level of indirection between a method call and the implementation of the method within a channel such that one channel can be substituted with another without affecting the caller. (SystemC term)

interface proper: An abstract class derived from class **sc_interface** but not derived from class **sc_object**. An interface proper declares the set of methods to be implemented within a channel and to be called through a port. An interface proper contains pure virtual function declarations, but typically contains no function definitions and no data members. (SystemC term)

interoperability: The ability of two or more transaction level models from diverse sources to exchange information using the interfaces defined in this standard. The intent is that models that implement common memory-mapped bus protocols in the programmers view use case should be interoperable without the need for explicit adaptors. Furthermore, the intent is to reduce the amount of engineering effort needed to achieve interoperability for models of divergent protocols or use cases, although it is expected that adaptors will be required in general.

lifetime (of an object): The lifetime of an object starts when storage is allocated and the constructor call has completed, if any. The lifetime of an object ends when storage is released or immediately before the destructor is called, if any. (C++ term)

lifetime (of a transaction): The period of time that starts when the transaction becomes valid and ends when the transaction becomes invalid. Because it is possible to pool or re-use transaction objects, the lifetime of a transaction object may be longer than the lifetime of the corresponding transaction. For example, a transaction object could be a stack variable passed as an argument to multiple *put* calls of the TLM1.0 interface.

loosely timed: A modeling style that represents minimal timing information sufficient only to support features necessary to boot an operating system and to manage multiple threads in the absence of explicit synchronization between those threads. A loosely timed model may include timer models and a notional arbitration interval or execution slot length.

master: This term has no precise technical definition in this standard, but is used to mean a module or port that can take control of a memory-mapped bus in order to initiate bus traffic, or a component that can execute an autonomous software thread and thus initiate other system activity. Generally, a bus master would be an initiator.

Comment [j1]: Or does it need to be re-introduced in the context of temporal decoupling?

method: A function that implements the behavior of a class. This term is synonymous with the C++ term *member function*. In SystemC, the term *method* is used in the context of an *interface method call*. Throughout this standard, the term *member function* is used when defining C++ classes (for conformance to the C++ standard), and the term *method* is used in more informal contexts and when discussing interface method calls. (SystemC term)

non-blocking: Not permitted to call the **wait** method. A non-blocking function is guaranteed to return without consuming simulation time or performing a context switch, and therefore may be called from a thread process or from a method process. A non-blocking interface defines only non-blocking functions.

object: A region of storage. Every object has a type and a lifetime. An object created by a definition has a name, whereas an object created by a new expression is anonymous. (C++ term)

OSCI bus, OSCI transaction level memory mapped bus protocol: A specific set of transaction attributes and their semantics together defining a transaction level protocol which may be used to achieve a degree of interoperability between PV models for components communicating over a memory-mapped bus.

Comment [j2]: TBD

OSCI initiator: A module, port or process that initiates an OSCI transaction. The initiator is responsible for initializing the state of the transaction object, and for deleting or reusing the transaction object at the end of the transaction's lifetime. In the case of the TLM 1.0 *get* interface and the response from the *transport* interface, the term initiator cannot be strictly applied, so the terms *caller* and *callee* should be used instead.

Comment [j3]: The intent is to encompass both TLM1.0 and TLM2.x transactions, but I'm afraid this is going to cause controversy in the case of 'get();'. See the definition of unidirectional interface.

OSCI interconnect component: A module or process that accesses a transaction object, but does act as an OSCI initiator or an OSCI target with respect to that transaction. An OSCI interconnect component may or may not be permitted to modify the attributes of the transaction object, depending on the rules of the OSCI bus. Arbiters and routers would typically be modeled as OSCI interconnect components, the alternative being to model them as both OSCI targets and OSCI initiators.

OSCI target: A module, export or channel that represents the final destination of a read or write OSCI transaction. For a write, data is copied from the initiator to one or more targets. For a read, data is copied from one target to the initiator. A target may read or modify the state of the transaction object.

OSCI transaction: A transaction as defined by the specific mechanisms of this standard. A system transaction may be represented by a single OCSI transaction or by multiple OSCI transactions. Each OSCI transaction is represented by a single transaction object. See *transaction*.

Comment [j4]: TBD. For an atomic transaction, all attributes become valid at the start time and do not change thereafter. TLM1.0 transactions are atomic.

phase: The period in the lifetime of a transaction occurring between successive timing points.

programmers view (PV): The use case of the software programmer who requires a functionally accurate, loosely timed model of the hardware platform for booting an operating system and running application software.

slave: This term has no precise technical definition in this standard, but is used to mean a reactive module or port on a memory-mapped bus that is able to respond to commands from bus masters, but is not able itself to initiate bus traffic. Generally, a slave would be modeled as an OSCI target.

Comment [j5]: Or does it need to be re-introduced in the context of temporal decoupling?

system initiator: The term system initiator is used to disambiguate the general usage of the term initiator from an OSCI initiator. See *initiator* and *OSCI initiator*.

system master: See *master*.

system transaction: The term system transaction is used to disambiguate the general usage of the term transaction from an OSCI transaction. See *transaction* and *OSCI transaction.*

target: This term has no precise technical definition in this standard, but is used to mean a component that is able to respond to transactions generated by an initiator, but not itself initiate new transactions.

temporal decoupling: The ability to allow one or more masters to run ahead of the current simulation time in order to reduce context switching and thus increase simulation speed.

timing point: A point in time at which the processes that are interacting through a transaction either transfer control or are synchronized. Certain timing points are implemented as function calls or returns, others as event notifications. Timing points mark the boundaries between the phases of a transaction.

transaction: An abstraction for an interaction or communication between two or more concurrent processes. A transaction carries a set of attributes and is bounded in time, meaning that the attributes are only valid within a specific time window. The timing associated with the transaction is limited to a specific set of timing points, depending on the type of the transaction. Processes may be permitted to read, write, or make themselves sensitive to attributes of the transaction. A transaction may be an atomic transaction or a complex transaction.

Comment [j6]: TBD. For an atomic transaction, all attributes become valid at the start time and do not change thereafter. TLM1.0 transactions are atomic.

transaction object: The object that stores the attributes associated with an OSCI transaction.

transaction level (TL): The abstraction level at which communication between concurrent processes is abstracted away from pin wiggling to transactions. This term does not imply any particular level of granularity with respect to the abstraction of time, structure, or behavior.

transaction level model, transaction level modeling (TLM): A model at the transaction level and the act of creating such a model, respectively. Transaction level models typically communicate using function calls, as opposed to the style of setting events on individual pins or nets as used by RTL models.

transactor: A module that connects a transaction level interface to a pin level interface (in the general sense of the word interface) or that connects together two or more transaction level interfaces, often at different abstraction levels. In the typical case, the first transaction level interface represents a memory mapped bus or other protocol, the second interface represents the implementation of that protocol at a lower abstraction level. However, a single transactor may have multiple transaction level or pin level interfaces. See *bridge*.

transport interface: The one and only bidirectional core interface. The transport interface passes a request transaction object from caller to callee, and returns a response transaction object from callee to caller.

unidirectional interface: A TLM 1.0 transaction level interface in which the attributes of the transaction object are strictly readonly in the period between the first timing point and the end of the transaction lifetime. Effectively, the information represented by the transaction object is strictly passed in one direction either from caller to callee or from callee to caller. In the case of **void put(const T& t)**, the first timing point is marked by the function call. In the case of **void get(T& t)**, the first timing point is marked by the return from the function. In the case of **T get()**, strictly speaking there are two separate transaction objects, and the return from the function marks the degenerate end-of-life of the first object and the first timing point of the second.

untimed: A modeling style in which there is no explicit mention of time or cycles, but which includes concurrency and sequencing of operations. In the absence of any explicit notion of time as such, the sequencing of operations across

multiple concurrent threads must be accomplished using synchronization primitives such as events, mutexes and blocking FIFOs.

valid: The state of [...] an object returned from a function by pointer or by reference, during any period in which the [...] object is not deleted and its value or behavior remains accessible to the application. [...] (SystemC term)