CABA/TLM-DT Transactors for the SoCLib virtual prototyping platform

Authors : Alain Greiner, Aline Vieira de Mello

This document is still under development.

A) Objetives

The main goal of the CABA/TLM-DT transactors is to integrate a small number of CABA (Cycle Accurate Bit Accurate) simulation models in a TLM-DT simulation environment. More precisely, we make the assumption that the shared memory interconnect is a TLM-DT model. This mixed mode simulation, where CABA & TLM-DT simulation models are cooperating in the same simulation environment can be useful to validate a CABA model (versus a pre-existing TLM-DT model), or simply to build an heterogeneous top-cell if, or the TLM-DT models, either the CABA models are not available for some hardware components.

B) General Principles

A CABA model for a VCI target is connected to a **VciTargetTransactor** component. Symetrically, a CABA model for a VCI initiator is connected to a **VciInitiatorTransactor** component. Both transactors have a CABA interface on one side, and a TLM-DT interface on the other side.

Each transactor behaves as a standard VCI initiator or VCI target TLM-DT component for the TLM-DT simulation environment. Each transactor contains a behavior() sc_thread modeling the PDES process (parallel Discrete Event Simulation) associated to the corresponding hardware component. According to the distributed representation of time in PDES, each transactor contains a local time variable, that is the time associated to the initiator or target respectively.

The VCI initiator is supposed to be able to initiate several simultaneous (parallel) VCI transactions. When there is several simultaneous transactions, the various transactions are identified by the TRDID field. The VCI PKTID field is not used Symetrically, the VCI target is supposed to be able to accept several simultaneous VCI commands, identified by both the VCI SRCID field (a given target can receive commands from all initiators), and the VCI TRDID field (each initiator can send several simultaneous transactions).

The modeling approach for the transactors respects the PDES distributed time representation : The clock signal CK of the CABA component is directly driven by the associated transactor component, depending on the local time evolution. If there are several CABA components in a given TLM-DT platform, there is no direct coupling between the CABA components (no sc_signal connecting two CABA component controled by two different transactors).

C) VCI Initiator Transactor modeling

All VCI commands received on the CABA interface are registered in a buffer indexed by the TRDID field. Each entry in this Pending_Transaction_Buffer contains a transaction status, a transaction time, and enough space to store a complete VCI TLM-DT transaction payload. As for any TLM-DT initiator, the behavior() sc_thread associated to the VciInitiatorTransactor is permanently running. It is descheduled twice per simulated cycle to drive the CK signal on the CABA interface. It implement the same look-ahead mechanism as any TLM-DT initiator to send NULL messages with a bounded period when there is no regular (Read or Write) VCI transactions. Regarding the VCI command, the behavior() thread read the command signals (cycle per cycle), on the CABA ports, and stores a complete transaction payload in the Pending_Transaction_Buffer. When the transaction is completed, the behavior() thread calls the nb_transport_fw() method on the TLM-DT port.

Regarding the VCI response, the interface nb_transport_bw() method updates the status of the proper entry in the Pending_Transaction_Buffer when a response is received on the TLM-DT port. This response is then transmitted (cycle per cycle) by the behaviour() thread on the CABA ports. If the VCI initiator can receive interrupts, the transactor provides an optional service of translation : When an IRQ is received on the TLM-DT IRQ port (by the interface nb_transport_fw() method) this information is registered, and the IRQ is transmited on the IRQ CABA port by the behaviour() thread.

D) VCI Target Transactor modeling

All transactions received on the TLM-DT port are registered in a 2 dimensions buffer, indexed by both the VCI SRCID, and VCI TRDID fields. Each entry in this Pending-Transaction_Buffer contains a transaction status, a transaction time, and a pointer on the transaction payload. As any TLM-DT target, the VciTargetTransactor component has a purely reactive behaviour : the behavior() thread is running only when a command is received, and keeps running until the last response to the last pending transaction has been transmited. When it is running, it is descheduled twice per simulated cycle to drive the CK signal on the CABA interface. Regarding the VCI commands received on the TLMDT port, the nb_transport_fw() interface method register the transaction in the Pending_Transaction_Buffer. The behaviour() thread will then send (cycle per cycle) the VCI command on the CABA port. Regarding the VCI responses received on the CABA port, the behaviour() thread updates the payload (cycle per cycle). When the transaction is completed, the corresponding entry in the Pending_transaction_buffer is cleared, and the nd_transport_bw() method is called on the TLM-DT port.

If the VCI target can send interrupts, the transactor provides an optional service of translation : When the behaviour() thread is running (following a VCI command, or NULL message reception) it polls the CABA IRQ port, and calls the nb_transport_fw() method on the IRQ TLM-DT port when the CABA value is modified.

E) Implementation

For both transactors, each entry in the Pending_transaction_Buffer can be in three different states :

- **EMPTY** : No registered transaction.
- **OPEN** : Command registered, waiting the response.
- **COMPLETED** : The response has been received

The main differences between the initiator & target transactors are the following :

- The Pending_Transaction_Buffer is larger for a target, because it is a two dimensionnal array, indexed by SRCID and TRDID. The target entry is simpler, because each entry contains only a pointer on the transaction payload.

- The behavior() thread is always runable for an initiator purely reactive for a target : It is not runable when the Pending_Transaction_Buffer is empty.