White Paper

An Introduction to the RapidIO® Bus Functional Models

By: Sam Fuller, President, RapidIO Trade Association
Maria Salieva, RapidIO Team Leader, Motorola Global Software Group

The RapidIO Bus Functional Models (BFMs) are modular behavioral C models of the RapidIO system interconnect architecture. The models are designed to provide a flexible tool to be used for the development, evaluation and verification of RapidIO interface products. At the interface level the models provide a "pin-level" interface in accordance with the 8/16 LP-LVDS (Parallel RapidIO) specification and 1x/4x LP(Serial RapidIO) physical layer specifications. The models also provide a two-way application programming interface (API) to the model environment for generating and responding to RapidIO transactions. The models can easily be integrated into event-based or cycle-based simulation environments.

The RapidIO models provide the following capabilities:

- Enable the creation of verification testbenches for RapidIO enabled devices
- Provide an accurate simulation of the whole RapidIO protocol
- Detect and report erroneous behavior
- Provide verification features such as error injection
- Enable RapidIO interconnect abstraction at different levels through a layered architecture approach

The models are written in ANSI C. This promotes their portability among different platforms. The model executables are supplied for the Sun Solaris 2.5.1 and RedHat Linux 7.1 platforms in the form of shared libraries and for Windows 2000 platform in the form of dynamically linked libraries. RapidIO model deliverables includes a "full" RapidIO model, a RapidIO physical layer model, a RapidIO transmitter-receiver (TxRx) model and a RapidIO Switch model:

The models support the whole RapidIO specification functionality, The RapidIO models can easily be customized by the user through the instantiation and initialization parameters to produce the desired external behavior and can also be scaled to provide the desired level of functionality.

The RapidIO model's general architecture and interfaces are shown in the Figure 1. The model architecture follows the RapidIO specification's layered approach while at the same time aggregating features for RapidIO design verification needs. The models provide interfaces that are used to connect the RapidIO models to local device models, memory models and cache models.

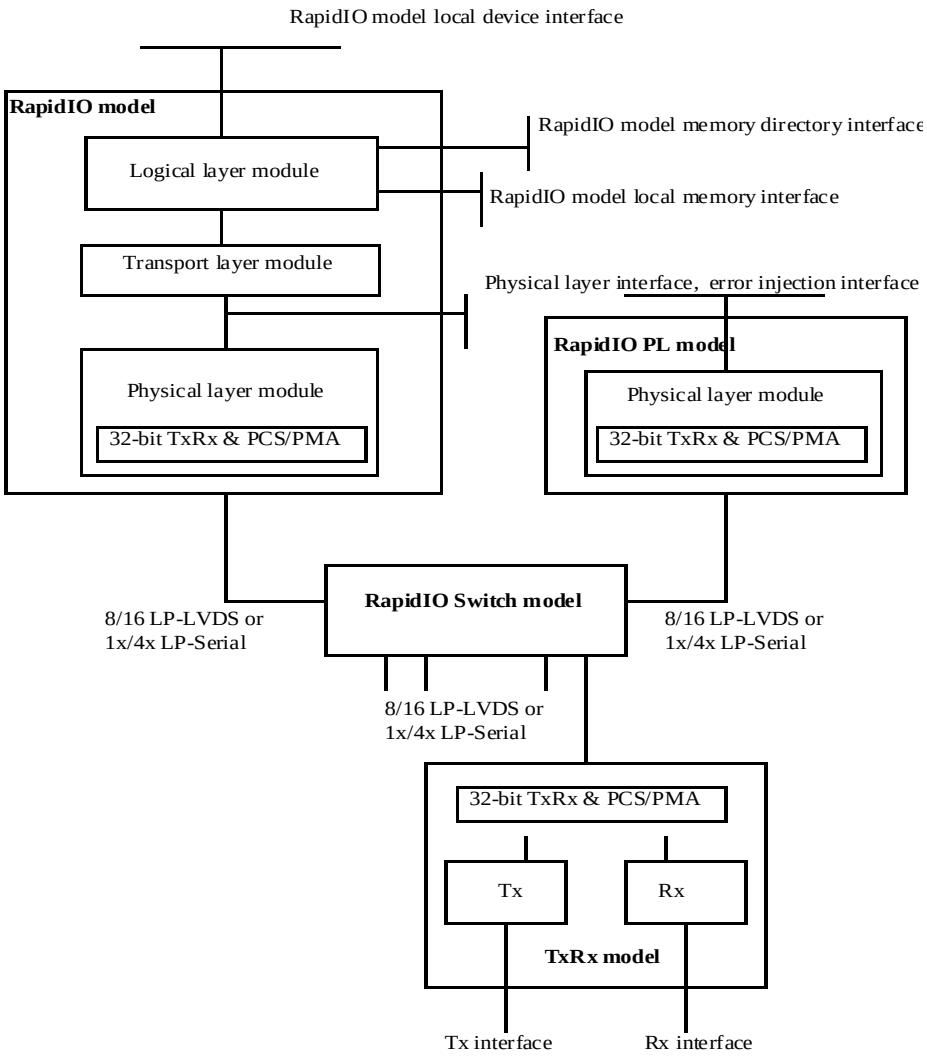

***Figure 1**: The Four RapidIO Bus Functional Models*

Figure 1 shows the four available RapidIO Bus Functional Models. These models are the Transmitter/Receiver (TxRx) Model, the Physical Layer (PL) Model , the Processing Element (PE) Model and the RapidIO Switch Model. The next four sections of this paper will examine the features and capabilities of each of these models.

The RapidIO TxRx Model

The RapidIO TxRx model implements the sending and receiving of packets and symbols on the RapidIO link. Physical layer protocol behavior is not enforced, so the user can use this model to create virtually any kind of erroneous situation. It is possible using the TxRx model to generate erroneous packets and control symbols. The TxRx model supports both 8/16 LP-LVDS and 1x/4x LP-Serial link types.

The RapidIO TxRx model has the following features:

- A consistent API for serial and parallel TxRx models
- Consists of separate receiver and transmitter components
- Allows detailed programming of the link protocol to support physical layer protocol verification (test of corner cases)

- Transmitting, embedding and receiving of 32 bit granules, symbols, packets (as a byte stream or requested via structure), code groups, characters, columns, compensation sequences
- Insertion of any kind of errors (packet/symbol structure, protocol, lost alignment, corrupt packets, symbols, characters)
- Model does not attempt to follow the RapidIO protocol (i.e. append EOP at the end of the packet, etc), just transmits what is requested and reports what has been received
- Configuration: Init state machine turn on/off option, 1x/4x, 8/16 LP-LVDS selection, status symbol generation on/off, compensation sequence generation on/off, compensation sequence rate

The RapidIO TxRx model provides the following interfaces (see Figure 1):

- The Tx (transmitter interface) is used to request the sending of packets/symbols/granules to the RapidIO link as well as for error injection.
- The Rx (receiver interface) notifies the user of granules, symbols and packets received on the link and of errors detected in their structure. It is also useful for low-level bus observability.
- The link interface is used to connect the RapidIO TxRx model to other RapidIO devices.

The RapidIO PL Model

The RapidIO Physical Layer (PL) model implements the functionality of the physical layer specifications and can be used for creating various kinds of transactions (including user-defined), as well as receiving incoming packets. Logical layer state machines are not implemented. The RapidIO PL model supports both 8/16 LP-LVDS and 1x/4x LP-Serial link types.

The main features of the PL model are:

- API is the same for the serial and physical models (except for error injection)
- Implements and handles all PL protocols as defined by the RapidIO spec
- Allows transmitting and receiving of arbitrary logical packets
- The model provides the following support
- Sending IO, Message, Globally Shared memory, Maintenance packets.
- The Model queues packets internally and sends them to the link
- Provides support for misaligned and multicast engines
- Reception of packets and delivery to the environment
- Automatic handling of link-level acknowledgement, retries, and error recovery
- Detects and reports all erroneous PL behavior on the link
- Error injection feature: corrupt packets, symbols, granules, characters, code groups before they go to link
- Monitor feature: reports all received symbols, packets, granules, characters, code groups
- Configuration: 1x/4x or 8/16 LP-LVDS, init sequence on/off, extended address width, number of buffer entries

The PL model interfaces provide the following capabilities (see Figure 1):

- The Physical layer interface is an analog of the RapidIO model local device interface. It is used by the environment to tell the RapidIO PL model to send packets to the link interface and is also used by the RapidIO PL model to notify the environment of received responses and incoming RapidIO packet requests.
- The link interface is used to connect the RapidIO PL model to other RapidIO devices.

The RapidIO PE Model

The RapidIO PE Model is a "full" RapidIO model that implements all of the currently defined RapidIO protocol functionality. This model may be used to represent a RapidIO Processing Element (PE), such as a processor-only PE, memory-only PE, processor-memory PE or I/O PE, etc. This model supports error injection into the outbound bit stream.

The RapidIO PE Model has the following features:

- Same API for the serial and physical interfaces
- Supports all logical layer, transport layer and physical layer protocols as defined by the RapidIO spec
- Provides interfaces for inbound transaction snooping (and detects collisions with outbound traffic), memory models, mailbox models, address translation
- Detects and reports all erroneous logical layer, transport layer and physical layer protocol behaviors
- Error injection features: Generation of corrupt packets, symbols, granules, characters, code groups before they go to link
- Monitor feature: Reports all received symbols, packets, granules, characters, code groups

The PE model interfaces provide the following features:

- The RapidIO model local device interface is used for requesting transactions from the RapidIO model. This interface is also used by the RapidIO model for snooping incoming (external) RapidIO requests and for notifying the local device of the completion of the requested transactions.
- The RapidIO model local memory interface is used to access the local memory of the RapidIO PE as a result of locally or externally requested transactions. Implementation of this interface is optional for the RapidIO model environment and would not be required if the modeled device contains no local memory.
- The RapidIO model memory directory interface is used to access the memory directory. The environment would implement this interface if it supports the GSM extensions and implements local memory.

The Physical Layer (PL) interface is used if the model is initialized to work in "PL" mode.

The error injection interface is used to emulate error injection over the RapidIO link.

The link interface is used to connect the RapidIO model to other RapidIO devices.

The RapidIO Switch Model

The RapidIO Switch model enables the user to connect several models in a simulation environment. The switch model supports both 8/16 LP-LVDS and 1x/4x LP-Serial link types.

Some important features of the switch model are:

- Implements and handles all transport layer and physical layer protocols as defined by the RapidIO spec
- Supports the development of a full RapidIO system model, consisting of multiple processing element and switch models
- Can model switches with up to 16 ports
- Each port separately programmable using port configuration registers
- Buffering at each input port for one packet
- Flow-through architecture
- Supports serial or parallel interfaces (although not in the same switch instance)

The RapidIO Switch model provides a configurable number of link interfaces that are used to connect the RapidIO Switch model to other RapidIO devices.

The RapidIO Models' Interface to the Simulation Environment

The RapidIO models may be used in event-based or cycle-based simulation environments. This means that each instance of a RapidIO model can be viewed as a parallel process, changing its internal state and producing output events (scheduling or changing output signals) in response to input events or a clock signal. The transition of the model to the new state and the production of output events/signal changes is made atomically, i.e. in zero simulation time.

To support this simulation style, the RapidIO model interfaces are designed to be two-way communication interfaces between the models and their environment. This means that the model provides a set of functions to the environment which should be used by the environment to request activities from the model or to notify the model of particular events so that transaction servicing can proceed (such as a snoop response ready event, memory data ready event, etc). To properly provide this support the model should be able to generate certain events and schedule signal updates to the environment. For example, the model may want to request memory access as a result of an external transaction servicing protocol, or to notify the environment of a received request packet (in case of the RapidIO PL model), or to schedule changes to the link interface signals. To provide this, the environment should supply callbacks to the model instance. The model will invoke these callbacks when it needs to generate events to the environment as a result of servicing transactions.

To avoid infinite recursion, the callbacks should not attempt to call the model internally. Instead, the event should only be scheduled in the environment to be seen by the environment at the next simulation cycle (or even later). This approach is shown in Figure 2.

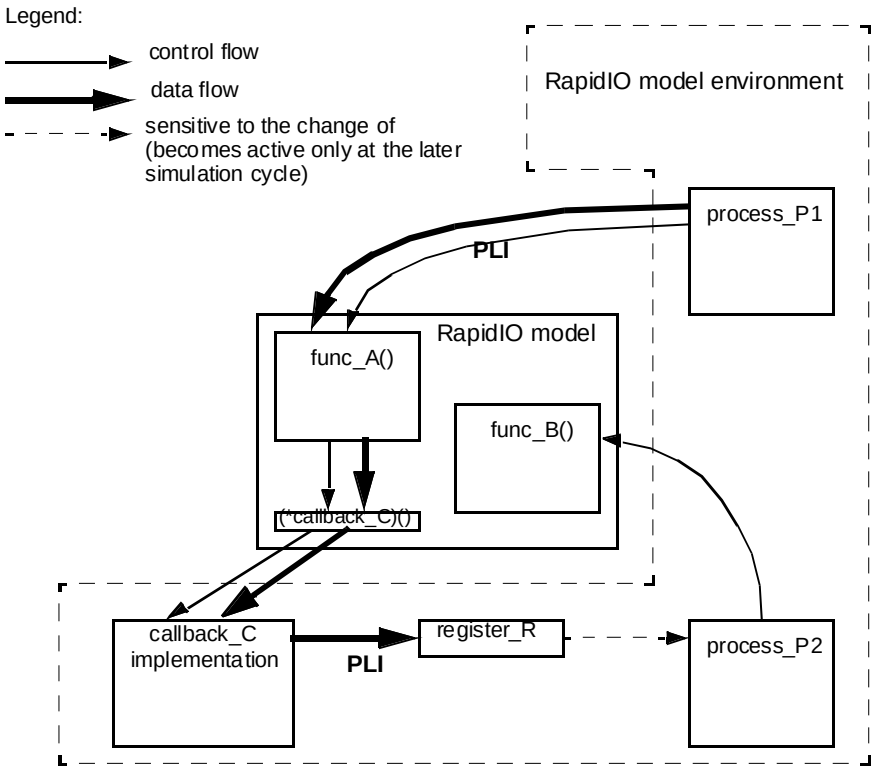

Figure 2: *RapidIO Models communication interfaces general approach (example for Verilog environment)*

The RapidIO models are written in ANSI C and are implemented as a library of functions. These functions access the state of the RapidIO model instances. To allow multiple model instances, the memory for instance data is allocated dynamically during instantiation of the model by the model instantiation function, which returns a handle to the instance data memory to the environment. It is the responsibility of the environment to store the handle and pass it when calling a RapidIO model. The environment should allocate the storage for the handle of each instance it wishes to create.

The RapidIO models are scalable to user requirements in the sense that there is clear correspondence between a callback function and the model functionality which requires this callback to be present. The circumstances for when the callback function should be executed are also clearly identified.

The interface timing of the RapidIO models is different for the RapidIO PL model, the RapidIO PE model and RapidIO Switch model. The RapidIO PL model needs to be synchronous because it interfaces directly to the link (either 8/16 LP-LVDS or 1x/4x LP-Serial) interface. The activities of the physical layer model are triggered by the input clock signal for the transmitter and by the input link interface clock signal for the receiver. The RapidIO PL model and the RapidIO Switch model are synchronous and perform their activities in response to clock events. The logical layer model interfaces are asynchronous and do not need to be linked to any clock signal.

Overview of the RapidIO Model Testbench

The RapidIO BFM testbench is a simple Verilog environment which wraps the RapidIO BFM model libraries inside Verilog modules, instantiates, initializes and connects the models and runs simple transactions on the RapidIO link. The testbench is intended to serve as a starting point for fast and efficient development of a custom verification environment.

Some of the functionalities of the RapidIO Models (such as address translation) are implemented as C code in the testbench. If changes to the behavior of the model implemented in C are required, it is necessary to modify the corresponding C code.

The RapidIO BFM testbench has been demonstrated to work with the ModelSIM SE/EE 5.4b simulator on the Windows 2000 platform and with VCS 6.0.1 on Sun Solaris 2.5.1, IBM AIX 4.3, RedHat Linux 7.1 platforms. Other simulators may be used, but the testbench has only been tested with the listed versions of the simulators.

The testbench consists of the following main parts:

- The Verilog environment which includes the Verilog modules that serve as wrappers for the models, including the Verilog top files
- The C environment which implements some of models' callback functions
- The PLI wrappers

An example of how the PE model is instantiated into the Verilog environment, along with sample code is presented in Figure 3.

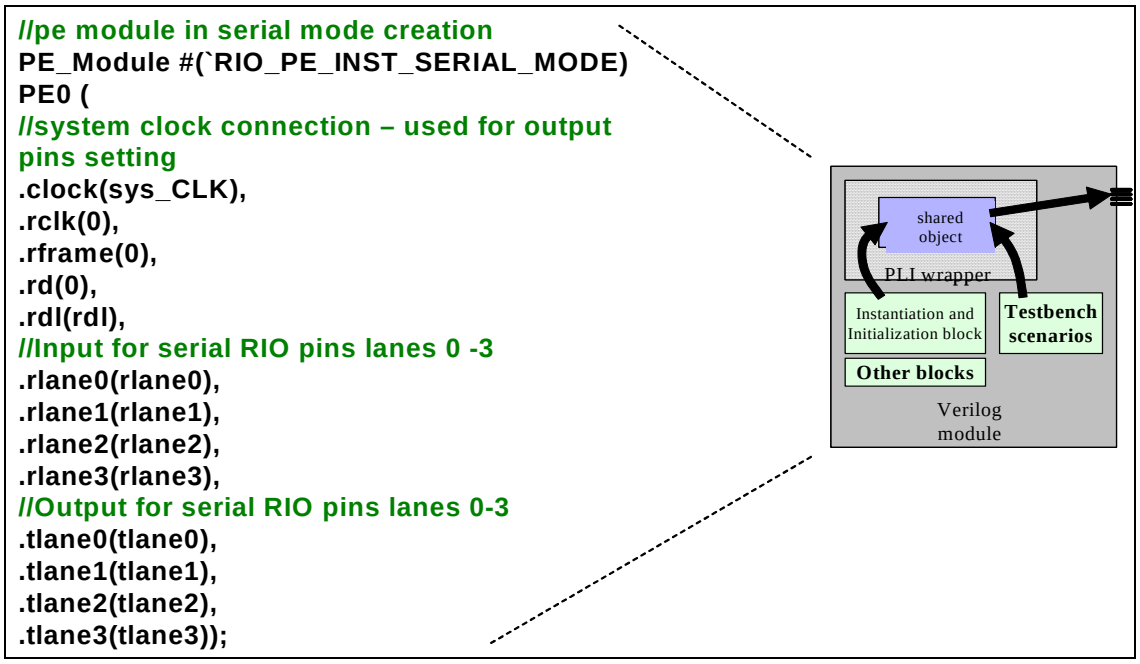

***Figure 3**: Example PE Instantiation*

In the testbench that is provided with the BFM models a PE model instance acts as the DUT. For a test of a user's RapidIO block the PE model would be replaced with the user's block.

The testbench uses some simple scenarios and includes a full list of Verilog versions of the model interfaces can be found in the wrapper code. The wrapper code includes support for all of the model's API routines. The routines have the same names as the C-routines except with a "$HW_" prefix added. Figure 4 below presents an idea of how the Verilog routines would be invoked.

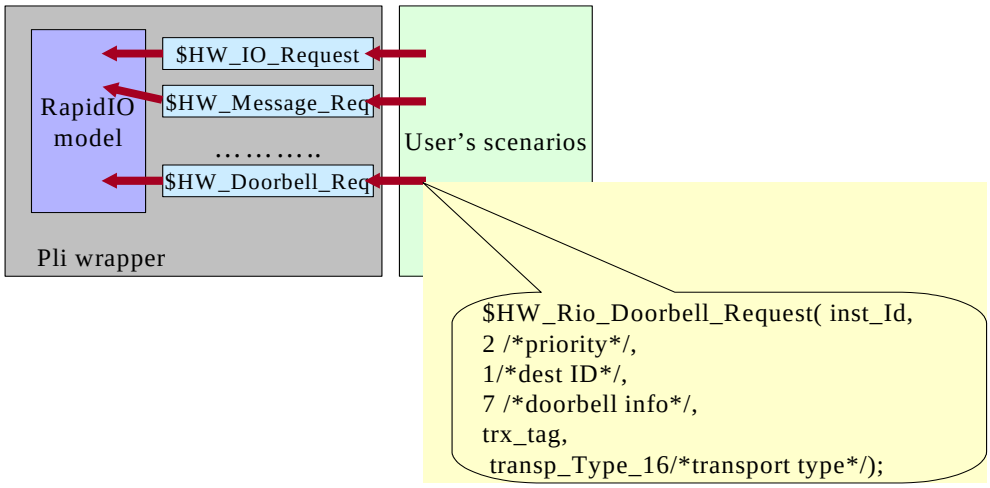

***Figure 4**: Driving the bus functional model through the Verilog Interface*

BFM Transaction Scripting

The BFM uses a scripting language for driving transaction in to the simulation environment and to the Device Under Test. The following code snippet provides an example of what these scripts would look like. These transactions are specified at for the Physical Layer (PL) model. The first two transactions would place data directly in to the payload of a IO request. The IO request is then sent twice in this example.

```
/*first DW of payload initialization*/
$HW_Put_Data_Pl (
`RIO_PL_IO_REQUEST_RQ_DATA /*name of the buffer*/, 0 /*offset in the payload*/,
32'h1E1E1E1E /*ms_Word*/, 32'h1F1F1F1F /*ls_Word*/);

/*second DW of payload initialization*/
$HW_Put_Data_Pl (
`RIO_PL_IO_REQUEST_RQ_DATA, 1,
32'h1E1E1E1E /*ms_Word*/, 32'h1F1F1F1F /*ls_Word*/);

/*Request for IO packet*/
$HW_Rio_PL_IO_Request( inst_Id, `RIO_IO_NWRITE, 1/*rq_prio*/, address, ext_Address,
0/*xamsbs*/, dw_size, offset, byte_Num, trx_Tag, transport_info );

/*Request for IO packet the payload will be the same as in the previous requested packet*/
$HW_Rio_PL_IO_Request( inst_Id, `RIO_IO_NWRITE, 1/*rq_prio*/, address, ext_Address,
0/*xamsbs*/, dw_size, offset, byte_Num, trx_Tag, transport_info );
```

Conclusion

The RapidIO BFM models have been used by nearly 20 companies to support the verification of RapidIO products. The set of PE, PL, TxRx and Switch models provide a complete set of features and functions for simulating and verifying RapidIO systems of almost arbitrary complexity. The various models are suited for varying levels of unit, block and system verification.

The TxRx or PL models are well-suited for unit or block level verification. The PL or PE models are best suited for chip level verification. The PE or Switch models are best suited for system-level verification.

The RapidIO BFM models, which are distributed in source code and object code form for several popular computing environments, are available to members of the RapidIO Trade Association through a sublicense agreement and payment of an annual support fee.

For more information on the RapidIO Bus Functional Models contact the RapidIO Trade Association at: gen_info@RapidIO.org.