Axis Review of SCE-MI Specification (Revision 1.9 dated March 20, 2002) December 12, 2002

In these comments RCC refers to the Axis emulation engine.

General comments about Software API

The software side interface should be a C API. The current specification is backwards. C++ can always call C functions. If C++ is desired C++ can wrap C functions. The spec provides more information in the C++ section. A user reading only the C section does not get all the information without also reading the C++ section. C will be easier to understand for more users so primary focus should be on C.

Section 2.1.1

Multi-threading relies on SystemC or the user to implement their own thread library. A C thread library should be included for users that do not want to use SystemC or write their own implementation. Axis is willing to donate a lightweight user level, non-preemptive thread library specifically designed for verification users with C test programs.

SCE-MI requires users to call the ServiceLoop() function to give processing time to SCE-MI. Axis prefers to first give to control to SCE-MI and have it activate the user program as necessary. This eliminates the need for the ServiceLoop() function. The combination of the ServiceLoop() and callback functions is confusing for users.

Hardware Interface

The clocking is the most restrictive portion of the specification because it contains hardware dependent information and far too much detail. RCC uses an event-driven algorithm with dynamic event signaling (just as in an event-driven software simulator). Users do not control the physical hardware clock in the emulator. Any discussion about the relationship between the testbench clock and the emulator clock should not be part of the spec.

RCC can run software tasks without the emulator advancing. There is no need for the concept of controlled and uncontrolled time. All references should be removed.

4.1.4 SceMiClockPort Macro

Section 4.1.4.1

The specified clock generation is hardware implementation dependent and cannot be part of the spec. RCC has different parameters to specify a design clock. Vendor differences in clock generation should be allowed. The generation of clocks is independent of SCE-MI. Axis proposes to remove this macro from the specification.

4.1.5 SceMiClockControl Macro

This macro uses a gated clock. Axis RCC engine is event-driven and this type of design is a poor practice in an event-driven engine because glitches can occur. It can also lead to mismatches between software simulation and emulation. RCC can handle using a clock as a data input, but introducing it results in extra risk that is not necessary. This information is too low-level for users developing models with SCE-MI. Axis proposes to remove this macro from the specification.

Clocking Summary

Both SceMiClockPort() and SceMiClockControl() are too detailed for a specification. The spec should allow for implementations that to not require the user to learn about the relationship between the controlled clock and the uncontrolled clock. Some implementations may require only a user clock for each interface. These two modules should be removed from the spec and left to the implementation.

The main purpose of the interface is to move data using SceMiMessageInPort() and SceMiMessageOutPort(). The spec should focus on providing these 2 interfaces. Adding a clock and user defined reset signal will make them general purpose and remove the requirement for the clocking macros.

Proposed Interfaces:

```
module SceMiMessageInPort(clk, reset, ReceiveReady, TransmitReady, Message);
  parameter     PortWidth = 1;

  input clk;
  input reset;  /* can be any user signal to reset the module */
  input ReceiveReady;
  output TransmitReady;
  output [PortWidth-1:0] Message;

module SceMiMessageOutPort(clk, reset, TransmitReady, ReceiveReady, Message);
  parameter     PortWidth = 1;

  input clk;
  input reset;
  input TransmitReady;
  output ReceiveReady;
  input [PortWidth-1:0] Messsage;
```

4.2 Infrastructure Linkage

All of the linkage can be instance-based. For RCC, everything can be determined by the SCE-MI implementation at time 0. There is no need for the infrastructure linkage information. Other vendors that cannot determine this automatically can use a configuration file or hardcode the information in C or verilog, but the configuration file should not be part of the spec.

Number of transactors – determined by instantiation

Transactor Name is the HDL instance name

Number of Message Input and Output Channels are determined by instantiation

Port Name is the HDL port name

Message Input or Output Port Width can be an instance parameter

Message Output Port Priority is not necessary since all ports must be serviced.

Number of Controlled Clocks is automatically determined by transactor ports.

Controlled Clock Name is determined by transactor port connections.

Controlled Clock Ratio is too low-level and should be transparent to the user. It is implementation dependent and should not be part of the spec.

Controlled Clock Duty Cycle and Phase is too low-level, user does not care.

Controlled Reset Cycles is too-low level, interface should reset automatically at the start or by a user reset signal.

4.2.1.1 Parameter File

There is no need for any centralized file in the specification; the interface can operate based on module instantiation alone. Individual vendors should be free to require such configuration files if necessary.

The corresponding C API calls related to the infrastructure linkage and parameter files should be removed from the spec.