International Journal For Technological Research In Engineering
Volume 2, Issue 7, March-2015 *ISSN (Online): 2347 - 4718*

MONITER BASED SPECIFICATION OF PCI OF INDUSTRY GRADED PROTOCOL

M.Vinod Gupta[1], T.Mounika[2], N.Sateesh Reddy[3], S.Rajesh Kumar[4], K.Prasanna Kumar[5]
[1,2,3,4]Student, [5]Asst. Prof
Department of ECE, Lendi Institute of Engineering and Technology, Vizianagaram, India

ABSTRACT: Bus protocols are hard to specify correctly, and yet it is often critical and highly beneficial that their specifications are correct, complete, and unambiguous. The informal specifications currently in use are not adequate because they are difficult to read and write, and cannot be functionally verfied by automatedtools. Formal sp ecifications, promise to eliminate these problems, but in practice, the difficulty of writing them limits their widespread acceptance. This paper presents a new style of specification based on writing the interface specification as a formal monitor, which enables the formal specification to be simple to write, and even allows the description to be written in existing HDLs. Despite the simplicity, monitor specifications can be used to specify industry-grade protocols. Furthermore, they can be checked automatically for internal consistency using standard model checker tools, without any protocol implementations. They can be used without modification for several other purposes, such as formal verification and system simulation of implementations. Additionally, it is proved that specifications written in this style are receptive, guaranteeing that implementations are possible. The effectiveness of the monitor specification is demonstrated by formally specifying a large subset of the PCI 2.2 standard and finding several bugs in the standard.

I. INTRODUCTION

VHDL style is based on writing the specification as a formal monitor. A monitor is an observer in a group of interacting modules, or agents which communicate via a set of protocol rules. Its main task is to flag agents when they fail to uphold the protocol. Writing the specification as a monitor enables the specification to be written as a list of simple rules, thus, allowing formal specification to be a relatively easy process. It also allows the specification to be checked "stand-alone" where no implementation needs to be written to verify the protocol. Furthermore, it results in a synthesizable specification which can be directly used in testing environments where cycle-based models are needed. Another direct use is for modeling environments when model checking an implementation. And despite its simplicity, a monitor specification can be written for "real" protocols such as the widely-used PCI local bus protocol. We also describe several highly effective debugging methods for monitor-style specifications. It is explained how certain requirements on the specification style discourages errors and how the debugging methods further ensure correctness and absence of contradictions. One highlight with a monitor specification is that debugging can be done on the protocol based on its internal consistency, before any implementations are designed. Furthermore, if two easy-to-check properties holds for the specification, it is guaranteed that the specification is receptive. Receptiveness guarantees that an implementation exists for the specification, and that there is no illusory freedom in the specification. On a practical level, these debugging methods found several problems in the official PCI protocol when they were applied to a specification of PCI.

The primary contributions of this paper are:
- The definition of a simple yet powerful specification style that is resistant to specification errors;
-Presentation of general specification debugging methodology, which does not require any implementations;
– A report on the successful application of the specification style and debugging methodology to PCI, and the resulting discovery of bugs in the protocol
– A theorem stating that the specification style together with a simple-to-check property, guarantees the receptiveness of a specification.

II. THE SPECIFICATION MONITOR

A. Description - the Specification Written as a Monitor

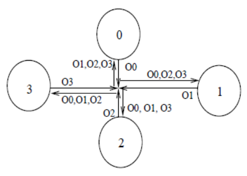

Fig. 1. The System View

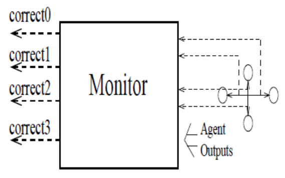

Fig. 2. The Monitor

A bus protocol specification can be viewed as a specification for a complete, closed system of agents using the bus. In Figure 1, agents 0,1,2,3 are using the bus and O0, O1, O2, O3 are the corresponding output sets. Because of the bus, the inputs for each agent are the outputs of other agents. (For agent 1, its inputs are O0, O2, and O3).A bus protocol specification dictates the behavior of all the outputs relative to each other. A monitor that checks the agents' compliance to the protocol at each execution step can be written. It is a machine with the agent output signals as its inputs, and boolean correcti signals as its outputs (figure 2). The monitor is such that as soon as an agent (or several agents) breaks the bus protocol, it singles out the erring agent(s) by making the corresponding correcti signal false. If correcti is true, agent i has upheld the specification so far and its current outputs also conform to the specification. If correct I is false, agent i

has broken a specification requirement currently or sometime in the past. Thus, correcti is “sticky”; once a rule has been broken, the corresponding correcti stayes false forever. The specification style is based on writing the specification as such a monitor.1 After all, the monitor must have exactly the protocol information to decide on agent compliance so it is natural for the protocol specification to be in the form of
a monitor; it differs from the conventional view of a specification only because it is an active machine as opposed to a passive documentation. The immediate benefits of this are the direct applications of such a specification.

B. For Model Checking a Single Implementation:
To verify a single agent implementation, one needs to create an environment for it, namely other agents on the bus. This is a non-trivial, tedious task. However with a monitor, an environment can be created without writing any implementation code. It does this by specifying which input sequences to the agent are correct according to the interface specification. Namely, one would model check the single agent by conditioning all the properties to be verified, with “if the interfacing agents have been correct so far according to the monitor”. For example, if p is the property to be model checked, and agents i and j form the environment, the property to be model checked is “correcti □ correct j _ p” where correcti and correct j correspond to the output signals of the monitor for i and j. The monitor and the condition in the model checking properties correctly constrain the inputs to the agent. This is an example of assume guarantee reasoning where the specification for one (or more) agent(s) is used to verify the implementation of another agent. This use of the monitor is very similar to what is described in [1]. As an execution example of this technique, Govindaraju used our PCI monitor to successfully verify a PCI controller implementation [10].

C. For Simulating Complete System Implementations:
In a testing environment, an interface monitor, if written in the language of the implementation, can be directly connected to a design and flag errors and correctly assign blame to the erring module in a system-level simulation. Since monitors can be written in synthesizable RTL, they can be used for tools that need cycle-level models instead of event-based simulation models such as formal verifiers or emulators.

D. Construction of a Monitor Specification
A further advantage of the monitor-style specification is that they are very easy to construct. First, it is noted that a specification is a list of rules. In particular, the official PCI specification is written that way. Thus, it is natural for the monitor to check for each of these rules independently. For clarity, these rules will be called constraints here. Here are some examples of PCI constraints, “trdy# cannot be driven until devsel# is asserted” “only when irdy# is asserted frame3 is deasserted”. Only the monitor is written by the specification writer. The agents in the figure are to be later implemented by someone else. As logic formulas, these can be written as follows; trdy □ devsel (if trdy is true, then devsel must be true) and prev_frame_ □ frame □irdy (if frame is true in the last cycle,then it must either be true in this cycle or if it’s not, irdy must be true). The goal was to keep the constraints as simple as possible to prevent the overall specification from getting complicated. When specifying PCI, it was found that the following constraint characteristics can be kept true, and the specification can still fully describe the protocol.

1. No CTL or LTL For the monitor specification, all of the PCI constraints can be written without using any CTL [6] constructs nor is knowledge of any linear time temporal logic (LTL) specifically needed. This is the basis for the claim that the specification style can be used with HDLs such as Verilog. In Verilog, the above example becomes,
(where correcti is initialized to 1)
If trdy && ! devsel) {correct=0};

2. No complex state machines only two types of very simple state machines were needed as auxiliary variables for the PCI constraints. One type is a event-recoding state machine which becomes true when a set event happens and remains true until a reset event occurs and is false otherwise. This is needed, for example, to “remember” whether the transaction is a read or a write. The second type is a counting state machine which starts to count after a set event, and stops counting either when a reset event happens or a limit is reached, whichever comes first.

3. Small time frames
With the help of the state machines described above, all of the constraints can be written with less than three time frames. Thus, the most complicated PCI constraint looks This property keeps the constraints compact. From a preliminary inspection of a more complex protocol than PCI, such as Intel’s Merced bus, properties 1 and 3 seem to hold for other protocols. Thus, a specification can be a list of compact constraints which are easy to maintain. And to construct the desired monitor, the constraints are directly used to determine the correcti’s. Assuming that each constraint constrains the behavior of only one agent, the constraints are grouped by the agent which they constrain. When the agent output signals make all the constraints of one agent true, the corresponding correcti is true; otherwise, the correcti is false. Thus, correcti is a conjunction of all the constraints specifying the behavior of agent i. The following is the assignment statement for correcti, where constraint j i then correcti = true, else correcti = false. Therefore, the monitor is a list of propositional formulas, auxiliary state variable assignments, and correcti assignments. There is no conversion of this to a state machine; this is precisely the code for the monitor.