Deadlock Prevention in the ÆTHEREAL Protocol *

Biniam Gebremichael[1], Frits Vaandrager[1], Miaomiao Zhang[1]**,
Kees Goossens[2], Edwin Rijpkema[2], and Andrei Rădulescu[2]

[1] ICIS, Radboud University Nijmegen, The Netherlands
[2] Philips Research Laboratories, Eindhoven, The Netherlands

Abstract. The ÆTHEREAL protocol enables both guaranteed and best effort communication in an on-chip packet switching network. We discuss a formal specification of ÆTHEREAL and its underlying network in terms of the PVS specification language. Using PVS we prove absence of deadlock for an abstract version of our model.

1 Introduction

The ÆTHEREAL protocol [2, 4] has been proposed by Philips to enable both guaranteed and best-effort communication in an on-chip packet switching network. The design of such a protocol, which has to meet all the functional and correctness requirements for best-effort and guaranteed traffic, is a difficult task. Typically, the designers play around with thousands of design alternatives before they commit to one. It is difficult to keep track of all design alternatives in a systematic way, and to make sure that the choices that have been made are consistent. Our contribution is that: (1) for one of the numerous design alternatives we produced a detailed, precise and highly modular formal model in PVS [1], and (2) within this model we were able to establish a key correctness criterion for the absence of deadlock. We believe that our work illustrates that formal specification languages, such as the typed higher-order logic supported by PVS, can be most useful to document complex designs, to help designers to clarify design choices and to resolve problematic inconsistencies in an early stage of the design process.

An extended version of our paper is available as technical report [3]. We refer to [3] for a much more detailed explanation of the ÆTHEREAL protocol, in particular of the routing algorithm that prevents deadlock. The report also describes in great detail how we formally modeled[3] the protocol in PVS and how we proved absence of deadlock for an abstracted version of the model. Finally, it evaluates our experiences in modeling the ÆTHEREAL protocol, discusses related work and points at interesting topics for future work.

* Supported by PROGRESS project TES4199, Verification of Hard and Softly Timed Systems (HaaST).

** Currently affiliated with Tongji University, China.

[3] The PVS sources are available at `http://www.cs.ru.nl/ita/publications/papers/biniam/noc/`.

2 The ÆTHEREAL Protocol

A network on chip, like any other network, is composed of nodes and edges between them. The nodes are classified into two groups depending on their position in the network, namely network interfaces and routers. Network interfaces are the service access points of the network. An interface that initiates a communication request is called an *active network interface port* (ANIP), and an interface that responds to a communication request is called a *passive network interface port* (PNIP). Routers provide the connectivity of the network. They do not initiate or respond to communication but just route packets from one interface to another. Each node in the network has a number of (bounded) buffers to store packets that have arrived and are waiting to leave.

Within a packet switching network it is relatively easy to offer a best-effort (BE) communication service, in which packets can be delayed to an arbitrary amount of time, and it is not possible to give a worst-case estimation. The main goal of the ÆTHEREAL protocol [4] is to also provide a guaranteed-throughput (GT) service within a network on chip. This is done by first reserving the resources (links) needed for the GT service for the entire duration of the service. The challenging part is to set up a new GT service using the BE services, which do not give any timing guarantee. Due to the limited buffer size (which are also shared by already running GT services) a deadlock scenario can easily occur, and the ÆTHEREAL protocol has to avoid such circumstance at all times. Once a GT connection is established, data may flow through this connection without difficulty. An important instrument for the establishment of a GT connection is the *slot tables*. Each routers is equipped with such a table, in order to book-keep which outgoing link is reserved for a given incoming link at a given slot time.

Establishing a GT connection starts when a source ANIP sends a BE `SETUP` packet to a destination PNIP. This `SETUP` packet will try to reserve all the links in the path that lead to the destination. The intention is that the GT service will follow the same path for its entire duration. The destination PNIP may not be connected to the ANIP directly, therefore the `SETUP` packet may have to pass through a number of routers, or the buffers of the routers as shown in Fig. 1. Each router has a separate unit (or buffer) called *reconfiguration unit (rcu)*, where the management of the slot table take place. During reservation request, an outgoing link is reserved if the link is free during the requested slot time, otherwise the request is denied. If the reservation is accepted, the `SETUP` packet is passed over to the next router, and the process goes on. If every reservation request is successful in all the nodes in the path (including the destination), then the destination PNIP sends a BE positive acknowledgment packet (`ACK`) to the source (the arrow with $**$ in Fig. 1). We say that the GT-connection has been established when the source receives the `ACK` packet. Subsequently, the GT service can start as scheduled. However if at some point in the path a node rejects the reservation request, the node will send a BE negative acknowledgment packet (`NACK`) to the source (the arrow with $*$ in Fig. 1). When the source ANIP receives `NACK`, it means (1) the GT-connection can not start, and (2) it has to unreserve the reservations it made. Note that the nodes between the source up to the node where the `SETUP` packet was rejected, do not know that the setup process has failed. For this purpose the ANIP sends a BE tear-down (`TDOWN`) packet to unreserve what has been reserved earlier. This `TDOWN` packet follows the same path as the preceding SETUP packet. Like SETUP packets, TDOWN packets visit every router on the path and update the slot tables accordingly.

One thing that may possibly go wrong during GT connection set-up is buffer overflow. This is handled by controlling the flow of packets locally (between adjacent nodes) and globally. As shown in Fig 1 local flow control is between adjacent nodes (or more specifically, adjacent buffers), and the global (or end to end) flow control is handled within ANIPs. For local control, the sender node maintains a local credit counter for every adjacent buffer. This counter records how much space is left in the receiver's buffer, and a packet is sent to this buffer only if it is not full. End-to-end flow control is introduced to prevent ANIPs from flooding the network. An end to end flow control counter is maintained locally by every ANIP in the network. Each time an ANIP sends a SETUP packet, its credit is decremented by one, and each time the ANIP receives an ACK or NACK its credit is incremented by one. Initially an ANIP has a credit which is equivalent to the size of the buffer in which acknowledgment packets are received in the ANIP (anip_ack_buffer). Thus, ANIPs may only send SETUP packets if they can accommodate the resulting acknowledgment packets.

A key idea to prevent deadlock in ÆTHEREAL is to have separate classes of buffers for system (SETUP, TDOWN) and acknowledgment (ACK, NACK) packets, and to ensure that there are no routing cycles within a buffer class. This separation is illustrated in Fig. 1 as a buffer dependency graph. The buffer dependency graph of a network on chip is defined to be a directed graph whose vertices are buffers and whose edges correspond to possible routings from one buffer to another. A key property that we proved for our PVS model of ÆTHEREAL is that there is no routing from an acknowledgment buffer to a system buffer, with the exception that in an ANIP a NACK packet may be routed to a system buffer as a TDOWN packet. Thus, if a path involves ANIP buffers then it may contain a cycle. But, as we will argue in the following section, even in this case no deadlock will occur.

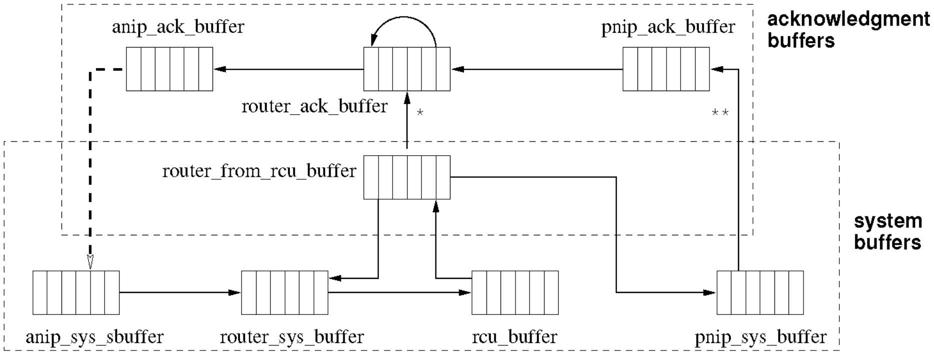

Fig. 1. Dependency graph between buffers

3 Deadlock Involving an ANIP

Communication in a network on chip takes place via synchronous transmission of packets from one buffer to another buffer. Each transmission is signaled by the advancement of a time slot [4]. The behavior of a complete network can be modeled conveniently by a state machine in which the states are the configurations of the network at a given time slot and the transitions correspond to the synchronous transmission of packets from one buffer to another.

A state is identified by the values of the following variables: (a) the content of the buffers, (b) local and end to end credits, (c) the slot tables, and (d) the time slot. Initially, all buffers and slot tables are empty. The local credit is equal to the buffer capacity it refers to, and the end to end credit is equal to capacity of the acknowledgment buffer of the ANIP. The time slot is zero.

The control transitions of the network can be structured as three sequential steps called *read*, *execute* and *write*. These three phases together constitute a single control transition in the state machine. We say that there is a transition (or `step(s1,s2)`), from a state `s1` to another state `s2`, if `s2` can be reached from `s1` by executing the three sequential steps. The set of reachable states is the set of all states that can be computed by recursive application of `step(s1,s2)`, starting from the initial state. We say that a reachable state `s` has a *deadlock* if there are a list of buffers `lb`, which are full in `s` and which form a cycle in the dependency graph.

In order to prove that there is no reachable state with a deadlock, we proceed by assuming the converse. Suppose that there is a state with a deadlock. This means that there is a list of full buffers containing ANIP buffers and this list forms a cycle. Moreover, this means that the system and acknowledgment buffers of the ANIP are full and yet there is an incoming packet from the network to this ANIP. But as explained above, the end-to-end flow control forbids such scenarios, because the ANIP could not have sent more packets than the capacity of its acknowledgment buffer. Formally, using PVS, we established (for an abstract version of our model) a number of system invariants which in combination imply that such a scenario will never arise.

References

1. J. Crow, S. Owre, J. Rushby, N. Shankar, and M. Srivas. A tutorial introduction to PVS. In *Workshop on Industrial-Strength Formal Specification Techniques*, Boca Raton, Florida, April 1995.
2. Om Prakash Gangwal, Andrei Rădulescu, Kees Goossens, Santiago Gonzãlez Pestana, and Edwin Rijpkema. Building predictable systems on chip: An analysis of guaranteed communication in the æthereal network on chip. In P. van der Stok, editor, Philips Research Book Series, chapter 1. Kluwer, 2005.
3. B. Gebremichael, F. Vaandrager, and M. Zhang. Formal models of guaranteed and best-effort services for network on chip. Technical Report ICIS-R05016, Radboud University Nijmegen, 2005.
4. Kees Goossens, John Dielissen, and Andrei Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22(5), Sept-Oct 2005. Special issue on Networks on Chip.