Università degli Studi di Catania

Dipartimento di Ingegneria Informatica e delle Telecomunicazioni

Corso di Sistemi Embedded per il Mobile Multimedia

Dario Fiumicello, Massimiliano Raciti

Implementation of a packet-level ACK mechanism for noxim

---

22 luglio 2008

Indice

1 Introduction

The purpose of our work was to implement a packet level ACK mechanism for noxim. In this way we can simulate more network scenarios. For example we can consider the traffic load relative to the ACK packet or the load due to lost ACK and packet retransmission (although ACK loss and retransmission was not yet implemented).

2 The current platform

Actually a node in noxim is divided in two elements: the router (TRouter.cpp) and the processing element (TProcessingElement.cpp). Either the router and the processing element have two methods called rxProcess() and txProcess() which are sensible to the clock front. Let's see what happen when these methods are called.

- *TRouter::rxProcess()*
  For each direction (North, South, West, East or Local) the router check if there is an incoming flit and if there is enough space in buffer. If these conditions are not verified the flit is discarded. If the flit can be accepted it will be put in the buffer awaiting for being transmitted. When a flit is accepted an acknowledge will be sent to the sending node by setting the corresponding signal (ack_rx[i].write()).

- *TRouter::txProcess()*
  For each direction the router verify if the corresponding buffer has some flit to be transmitted. If it's so it gets the flit and verify if it is an HEAD flit. In this case it tries to reserve a channel for it. Then, if the flit got a channel, it will be forwarded through it and the flit will be popped out from the buffer. When the tail flit cross the router the reservation will be freed and the channel became available.

- *TProcessingElement::rxProcess()*
  Every time the rxProcess() is invoked it checks the req_rx signal value; if it is equal to 1-current_level_rx (this is the Alternate Bit Protocol) it means that a new flit is on the wire. The processing element simply read the flit, update the value for the current_level_rx and send an ack

by putting the ack_rx signal to current_level_rx. The processing element won't process anything (quite a contraddiction!)

- *TProcessingElement::txProcess()*
  When txProcess() is called it checks if it can shot a packet using the canShot() method; canShot() return true based on the traffic generator function. canShot() accept a packet as a param and fill that packet with data if it can be shoot. Once created the packet it is pushed into the packet queue, ready to be transmitted. At this time, if the packet queue is not empty the processing element get the next flit of the front packet and forward it down to the router, setting the req_tx signal according to the Alternate Bit Protocol.

3 The packet-level ACK implementation

To implement a packet level ack protocol we modified the txProcess() and rxProcess() methods of the TProcessingElement class and the txProcess() method of the TRouter class. Then we added a new flit type (FLIT_TYPE_ACK) wich is the only flit that compose an ACK packet. The modification to the TRouter::txProcess() method was necessary to let the router handle this new kind of flit. Finally we modified the TPacket structure adding a new field that contains the sequence bit for that packet (so future implementation can use this bit to implement retransmission or flow control protocols, like the stop and wait). The sequence bit for a packet is determined per each destination, so an array of N bits is required to store the previous sequence bit (N is the number of tiles in the NoC).

- *TProcessingElement::rxProcess()*
  Instead of processing nothing, the rxProcess() now checks for the received flit type and if it's a FLIT_TYPE_TAIL it put an ACK in the packet queue for the last packed received, so that the ACK will be sent back to the sender as soon as it reachs the queue head. If the received flits is a FLIT_TYPE_ACK then the processing element simply print that it has received it (if verbose mode is used).

- *TProcessingElement::txProcess()*
  The only changes in this method are about setting the sequence bit in the packet by negate the previous sequence bit for that destination.

- *TRouter::rxProcess()*
  The FLIT_TYPE_ACK is either an HEAD and a TAIL flit. When the router sees an ack flit it must try to reserve it a channel as if it was an head flit and it must deallocate that channel after transmission, as if it was a tail flit. To obtain this behaviour we must put the condition that if te router sees a head flit OR an ack flit it has to try to reserve a channel and if it sees a tail flit OR an ack flit it must release that channel.

4 Open issues and future works

Actually the packet level ack mechanism does nothing! It simply respond with an ack when it receives a tail flit (the end of a packet). In a more realistic scenario ack packets can be lost or packets can be corrupted, so future works may regard the implementation of random lost or corruption of packets and retransmission protocols. For example a stop and wait protocol may be implemented, but we must consider the following issues: when the Processing Element must send the ack in a stop and wait protocol if the sender send a packet and stop sending anything until it receive the ack, deadlock conditions may occur if the ack is put at the end of the tx queue. Suppose that A send a packet to B and at the same time B send a packet to A: when B receive the packet from A it puts the ACK back in the tx queue (so do A), but the tx queue is locked because B is waiting for ACK from A, which is locked too waiting for the ACK from B! This issue implies that the ACK must be sent as soon as the tail flit is received, so it should preempt other packets. By the way, packets already being sent must complete before the ack is transmitted, so the preemption must be per packet and not per flit. How many queues? If the transmission is locked until the ack is received, other packets destined to other tiles are blocked too! It is necessary to implement a queue for each tile or to use a protocol that allow the transmission of packets that are destined to other tiles.