T-79.179 Parallel and Distributed Digital Systems Project / Maria reachability analyzer

The project can be thought of as consisting of two major parts: 1. modeling a simple system, and 2. answering some questions regarding the modeled system. The grade will be based on all questions, meaning that if some questions are left unanswered, the grade will suffer accordingly.

Return the assignment by email, in plain text, containing the models, answers to the questions, and transcripts of the necessary analyzer runs. The deadline for the assignment is May 9th 2005. Address the emails to Jukka.Honkola@hut.fi with subject line starting with T-79.179. The submissions will be acknowledged. You can return multiple submissions; the grade will be based only on the latest one.

The Maria analyzer is installed in computing centre's Debian workstations and kosh.hut.fi. The command "use maria" performs the needed initializations. The documentation for the tool is available from the course's webpage http://www.tcs.hut.fi/Studies/T-79.179/ .The analyzer is available in http://www.tcs.hut.fi/Software/maria/ .

The modeled system is SpaceWire (SpW), a standardized interconnection network for use aboard spacecraft. The rationale behind the system is to provide a specification of the interconnection network for the manufacturers of space components which they can implement in the equipment. A short description is given below. Further information is available in the standard, which can be found in the course's webpage.

What you will have to do:

1. **Modeling the *initialization* of a single link of the SpaceWire network**. You do not need to model any handling of data (N-chars) except which is necessary to the modeling of the control (some error situations). Read carefully the text below, and check relevant parts of the pages 60-74 in the standard. The model should include the two ends of the link, that is, two separate link control state machines. It will be a good idea to spend some time to think about which features and signals need to be represented and in which way. If (when) you decide to leave something out, remember to justify shortly the decision.
   - **(a)** Model the communication with small (start with e.g. size 2) buffers. This means that the sent messages arrive in a buffer. The recipient performs actions based on the first message found in the buffer.
   - **(b)** Model the communication without buffers. This means that the message will be lost if the recipient will not accept it. Thus, for each message sent there are basically two cases: either the recipient accepts it (changes state), or the recipient rejects it (does nothing).
2. **Timers.** Timers are usually modeled in Petri Nets by allowing the timers to expire at any point between setting and resetting them. In the state machine, timers can be thought as being set on entry to the state which has timed transitions, and reset when a transition out of the state happens. What are the pros and cons of this method?
3. **Deadlocks.** Are there any deadlocks in the two models? Present a transcript of the analysis run (use for example the command `script` in unix/linux). If there is a deadlock, explain the reasons for it.
4. **Properties.** Verify the property "Eventually both ends of the link will be in state RUN" with the Maria analyzer and explain the reasons for the possible counterexample. Present a transcript of the analysis run. The property can be expressed in LTL (Linear time Temporal Logic) as a formula of the following form: <>("interface 1 in state run" && "interface 2 in state run") where, informally, the "<>" (eventually) operator specifies that the following formula has to be true in some state in the future.

A short description of the SpaceWire protocol and architecture

The SpW standard specifies the functionality of the network interface, used addressing scheme, and routing. For the network interface, functionality corresponding to the OSI physical and data link layers are specified. The network consists of nodes (end systems) and routers. End systems contain one or more interfaces while routers contain several interfaces and a routing table. The direct connection between two interfaces is called a link.

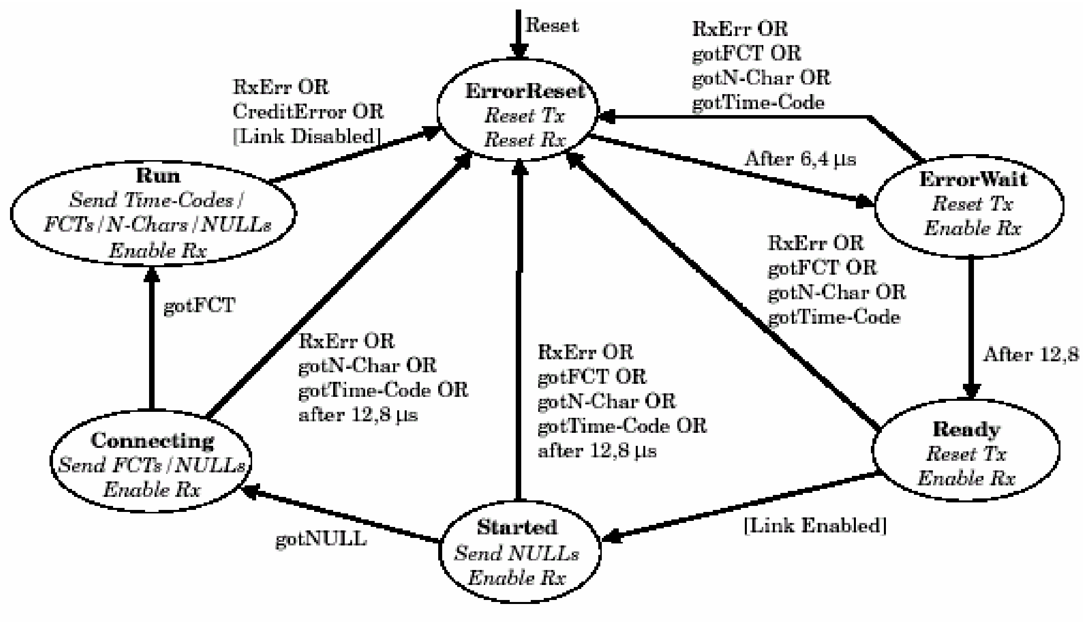
The SpaceWire protocol is a very simple link layer protocol. It has error detection but no means for error correction or retransmission. However, the network has been found to be very reliable, so error correction would probably only generate unnecessary overhead. Basically, the sender sends a packet and forgets about it. If an error occurs, the receiver will receive the packet up to the location of the error. The packet length is unbounded, but applications are recommended to avoid very long packets. The data is transmitted as characters.

The protocol is divided into four levels, namely character, exchange, packet, and network level. The character level describes the structure of characters, exchange level describes the transmission of data over a single link, packet level describes the structure of the packet, and network level describes the addressing and routing necessary to deliver packets to correct host.

The characters are grouped into N-chars (normal) and L-chars (link). N-chars are the characters that are passed on to the packet level. Data characters, end of packet (EOP) and error end of packet (EEP) are N-chars. All other characters are L-chars. The link control functions are performed in the exchange level. The main functions of the control plane are link error detection and subsequent connection re-establishment, and flow control.

The types of errors that are detected after the link is initialized are: parity error, credit error, disconnect error and escape error. Parity error occurs when the parity bit does not match the received character. Credit error is related to flow control. An interface keeps track of the characters that it is allowed to send to the other end. The allowed number of characters is referred to as credit. An interface sends flow control tokens (FCTs) to the other end to signal that it can receive more characters. An FCT allows the sending of eight characters. Upon receiving an FCT, the sender adds eight to its credit. Furthermore, there is an upper limit to the credit, namely 56. If the upper limit is exceeded or the interface receives characters when it expects none, a credit error has occurred. Disconnect error occurs when there is no traffic on the link for 850ns. If there are no data or control characters to send, the interface will send NULL characters as keepalive messages. An escape error occurs when an escape character is received unexpectedly. All the errors will result in connection re-establishment.

The exchange level operation is governed by state machine shown below. The state machine shows the link initialization sequence and possible errors. In essence, the initialization proceeds by gradually establishing the connection. First, there are timed transitions which guarantee sufficient time for the interfaces to reset. After that, the initialization proceeds by first checking that something can be received (NULL) and then waiting for an FCT to enable sending to the other end.



RxErr = Disconnect error OR Parity error OR Escape error (ESC followed by EOP or EEP or ESC).

NOTE Disconnect error only enabled after First Bit Received. Parity Error, Escape Error, gotFCT, gotN-Char, gotTime-Code only enabled after First NULL Received (i.e. gotNULL asserted). Thus RxErr OR gotFCT OR gotN-Char OR gotTime-Code is really RxErr OR (gotNULL AND (gotFCT OR gotN-Char OR gotTime-Code)).

Figure 20: State diagram for SpaceWire link interface

The routers use wormhole routing, where the forwarding of packet is started as soon as the address can be determined. Thus there is no need for buffering and the routers can be simpler. However, this type of routing can lead to deadlocks if there are routing loops. In any case, the SpaceWire network will probably be fairly small and also have a fairly stable topology. Therefore the routing loops should be easily avoided.