SAE Paper No: 2002-01-0445

Cycle-Accurate LIN Network Modeling and Simulation

Qian Chen, Yibing Dong and Salim Momin
Virtual Garage, Motorola Inc.

ABSTRACT

LIN (Local Interconnected Network) is a serial communications protocol that supports the control of mechatronic nodes in distributed automotive applications. This paper discusses LIN network modeling and simulation based on a token-based and event driven simulation platform. The complete LIN network features are modeled in the behavior level. The simulation is time-accurate and it provides system information, such as CPU load, bus utilization and message latency time. It can also simulate the scenarios such as network sleep and wakeup, switch event and error message. This LIN network simulation model can be integrated with CAN network simulation model for a complete vehicle network simulation.

INTRODUCTION

LIN is a low cost network which complements the existing portfolio of automotive multiplex networks. It is starting to become popular in the automotive industry. One major problem in the LIN network development is that before the hardware network is built, there is no way for the network designer to collect enough accurate system information of the network, such as CPU utilization, bus utilization and system response time. The system response time is especially a problem in the case that we need to consider some system issues, such as network wakeup, error message handling and gateway operation. Building the hardware network is costly and time consuming and any change in the design may require a complete rebuilding of the hardware network. Moreover, it is often too late to find the design flaws after the hardware network has been built and completely tested. The situation can be worse for the LIN network design in the sense that normally LIN network need to be connected to the CAN networks. It means that the thorough tests of the LIN network design may not be performed until both the LIN network and CAN network are built. Obviously, a simulation environment that can provide accurate information of the system performance for the LIN network design before any hardware is available will be a great help in the LIN network development.

LIN is a sub-network with speed up to 20k bit/s, wire length less or equal 40 m and recommended maximum 16 nodes. The cost for LIN is low since it uses single wire, its silicon implementation is based on common UART/SCI interface and it does not require quartz or ceramic resonator in the slave nodes. LIN uses a single-master / multiple-slave concept which guarantees the latency times for signal transmission (assume that there is no error). The LIN network structure can be described briefly with the following Figure 1.1:

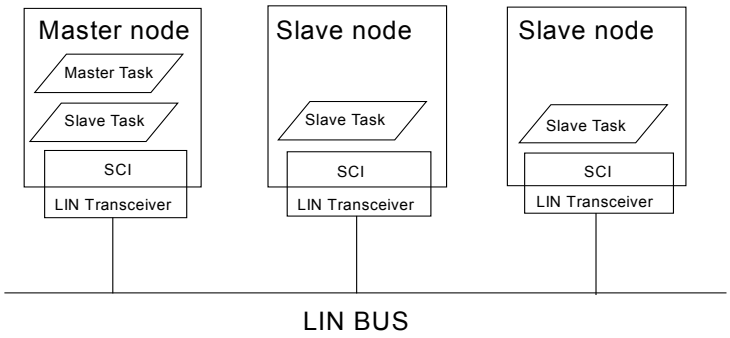


Figure 1.1 LIN network structure

LIN message has a fixed frame format with configurable data length (2, 4 or 8 bytes). A message frame includes two parts: header and response. The header consists of three fields: synchronization break, synchronization field and identifier fields and the response consists of several (2, 4 or 8 bytes) data fields and one checksum field. The header carries synchronization and identifier information from the master task to the slave tasks. Only the master task can send out a header. The response carries the data information. It is sent by the slave tasks from either a master node or a slave node. If the slave task fails to respond to a message header, the master task will transmit a new message header after a maximum time-out. The format of LIN message frame can be described with the following Figure 1.2:

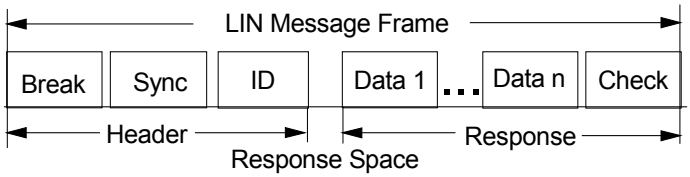


Figure 1.2 The LIN Message Frame

LIN NETWORK MODEL

The LIN network modeling is based on a simulation platform that is token based and event driven. The hierarchy of the LIN network modeling is as follows: The network model consists of the models of master node, slave nodes and LIN bus. The model of each node consists of the partition of hardware architecture and software architecture, which model the hardware and software behavior of the node respectively. The hardware architecture includes the modeling of the hardware blocks that are related to LIN message transmission, such as CPU, SCI and timer. The software architecture includes the modeling of the software tasks and the relationship among various software and hardware tasks. Furthermore the flowchart and cycles information of each software task are profiled and mapped into the software task model so that our LIN network simulation is cycle accurate. The LIN bus is modeled as a monitor of the message traffic in the bus. The hardware and software architectures of the master node and the slave nodes are different and they will be discussed separately in the following sections. Following is an example of the LIN network simulation model.

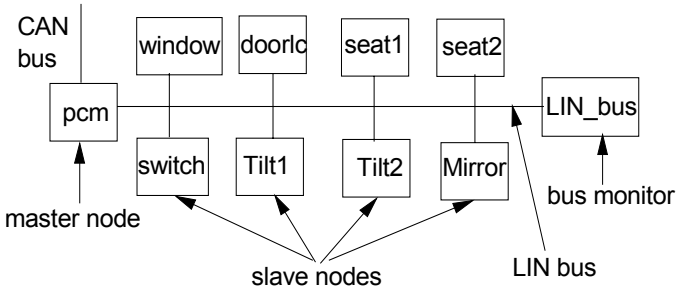


Figure 2.1 LIN network simulation model

HARDWARE ARCHITECTURE MODELING

The hardware architecture is one basic component in our hierarchical model structure. The LIN hardware architecture modeling include the modeling of the master node and the modeling of the slave node.

MASTER NODE HARDWARE MODELING

In the master node model, we have the hardware models of CPU, SCI and timer. These are the basic hardware blocks required for the operation of a LIN master node. The SCI is the physical layer for the LIN master node and a timer is required for the scheduling of message transmission and the detection of slave-no-response time-out. More hardware blocks can be added to the model if required. The hardware architecture for the master node model is described with the following Figure 3.1.

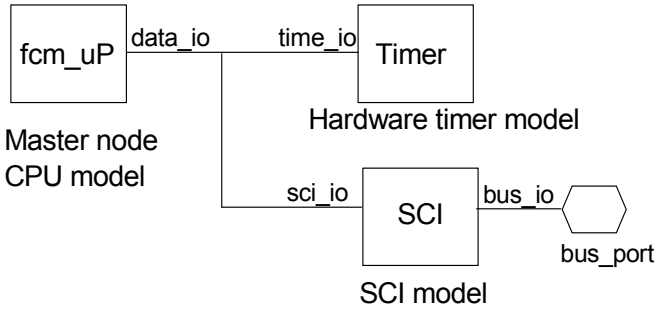
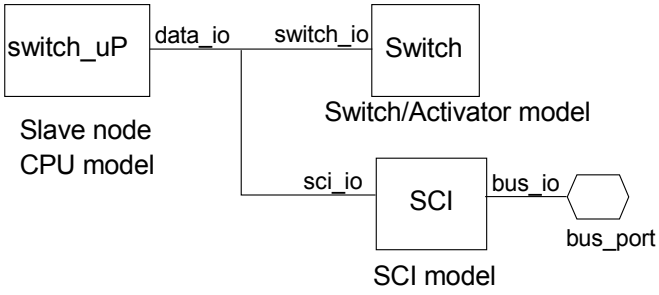


Figure 3.1. Master node hardware modeling

SLAVE NODE HARDWARE MODELING

The hardware architecture for the slave node model is described with the Figure 3.2. Here, compared to the master node model, we have a model block for the switch/activator in addition to the CPU and SCI models but we do not have a timer model block. That is because the switch and activator are normal for the functionality of a slave node, while a timer is not so critical for a slave node except for the detecting of bus idle time out. (This functionality is not simulated in our model since the normal simulation time is not long enough to observe this time-out.) The switch/activator block will generate the event stimulus for the simulation and the response to the switch stimulus, which enable us to check the latency time for the switch response.



.

Figure 3.2 Slave node hardware modeling

SOFTWARE ARCHITECTURE MODELING

Besides the hardware architecture, the model also needs the software architecture. The software architecture defines the relationship among various software and hardware tasks that simulate the functions of the LIN protocol. As mentioned before, our LIN network modeling is based on a simulation platform that is token based and event driven. Therefore, each software task block and hardware block will communicate with the transmission of tokens. The software architecture defines the type of the message to be transmitted and the route of the transmission. By doing so, the software architecture describes the relationships among different software tasks and hardware blocks. The software architecture modeling also includes the master node modeling and slave node modeling.

MASTER NODE SOFTWARE MODELING

The software architecture for the master node is described with the following Figure 4.1. As stated before, the master node includes a master task and a slave task. The modeling describes the type of messages to be transmitted (not shown in the Figure 4.1) and the route of transmission among the slave task, master task, timer hardware block and SCI block. Hence, the relationship among these two tasks, the timer model and the SCI model is clearly defined in the software architecture.

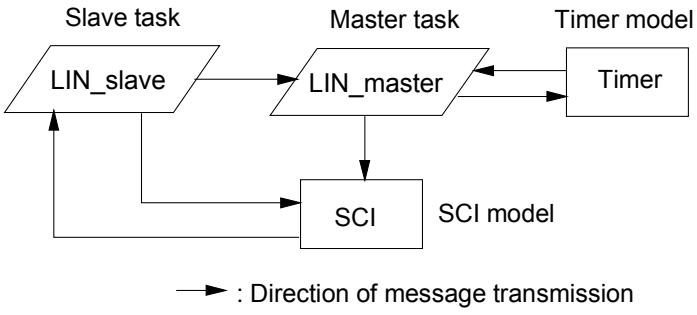


Figure 4.1 Master node software modeling

SLAVE NODE SOFTWARE MODELING

The software architecture for the slave node is described with the following Figure 4.2. The basic concept is the same as that of the master node. The software architecture defines the relationships among the slave task, switch/activator block and SCI block. The difference is that it consists of different component blocks. For example, the slave node only has slave task. In our model, it does not consist of the timer block while it includes a block for the switch/activator (refer to the slave node hardware architecture model). Which block is included in the architecture is determined by the specific application need.

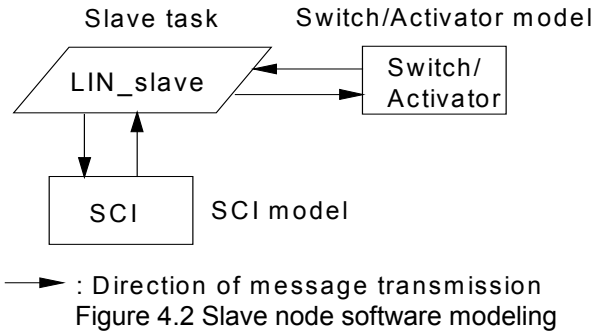


Figure 4.2 Slave node software modeling

SIMULATION AND RESULTS

In order to become a time-accurate simulation model, this LIN network model needs to include not only the hardware architecture and software architecture but also to simulate the software processing with accurate timing information. Therefore, in the simulation model we incorporate the timing information from the Motorola LIN drivers running in Cosmic M68HC12 simulator to make it cycle-accurate. In order to do so, first we need to profile the flow chart of the Motorola LIN drivers and run the LIN drivers in the simulator to profile the desired timing information. This timing information is incorporated in the programming of the software modeling of the LIN network to provide cycle-accurate timing penalty for the CPU. So the simulation result can show the CPU running time for each task and total CPU utilization. Furthermore, each software task can be assigned a priority such that high priority task can interrupt low priority task just as the how the real software works. Therefore, this simulation not only provides the CPU timing information as accurate as we can, but also provides the software processing delay for the message transmission.

The following Figure 5.1 shows an example activity chart for the LIN network simulation result. Each small mark represents an activity for the corresponding hardware or software block. Either it is processing a task or transmitting a message. This activity chart clearly shows the operation of the LIN network. For example, from this chart we can see that our simulation model simulates the following LIN network scenarios:

- How the LIN network wakes up from the sleep mode when a switch is pressed.
- How the LIN master node sends the message header and the slave node sends the response.
- How the slave-no-response time out works when the slave node fails to respond to message header.
- How the switch command is transmitted to the activator and how long is the latency time.

Also the simulation result can show us the information of CPU bandwidth and bus utilization which will be very useful when we analyze the penalty of LIN network operation in a LIN-CAN gateway.

Besides this activity chart, the simulation also provides the detailed information about each node and each message. For example, when does an error occur and what kind of error it is. Also, with a convenient GUI, the user can get the information such as: what is the maximum latency time for each message, how many messages are transmitted and received by each node and what is the dynamic moving average CPU load of each node. Finally, by clicking a button, the user can get an automatically generated report of the statistical data of each node and each message. Figure 5.2 gives a brief screenshot of these results and analysis.

As mentioned before, LIN network is usually used as a sub-network of CAN network. Several LIN networks and CAN networks will work together and messages and signals are exchanged among these networks. One important advantage of our LIN network simulation model is that it can be connected with our CAN network simulation model to perform the complete simulation of several networks as desired.

CONCLUSION

This paper presents the modeling and cycle-accurate simulation of LIN network based on a token based and event driven simulation platform. The simulation can provide accurate system information of the network to the network designer far before any hardware is built. It can also provide the module builder information of the network impact on the performance of each module. Furthermore this LIN network simulation model can be connected with our CAN network simulation model to perform a complete multi-network simulation.

REFERENCES

[1] LIN consortium, “LIN Protocol Specification, Revision 1.2”, November 17, 2000.

[2] LIN consortium, “LIN API Recommended Practice, Revision 1.2”, November 17, 2000.

[3] Wense, H., "Introduction to Local Interconnect Network", SAE World Congress, Detroit, March 2000. Document No. 2000-01-0145, SAE press.

[4] Motorola Inc., "LIN12 Driver User's Manual, Rev. 1.2", January 12, 2001.

CONTACT

Dr. Qian Chen
Virtual Garage, Motorola Inc.
41700 Six Mile Road
Northville, MI 48167
Email: Qian.Chen@motorola.com

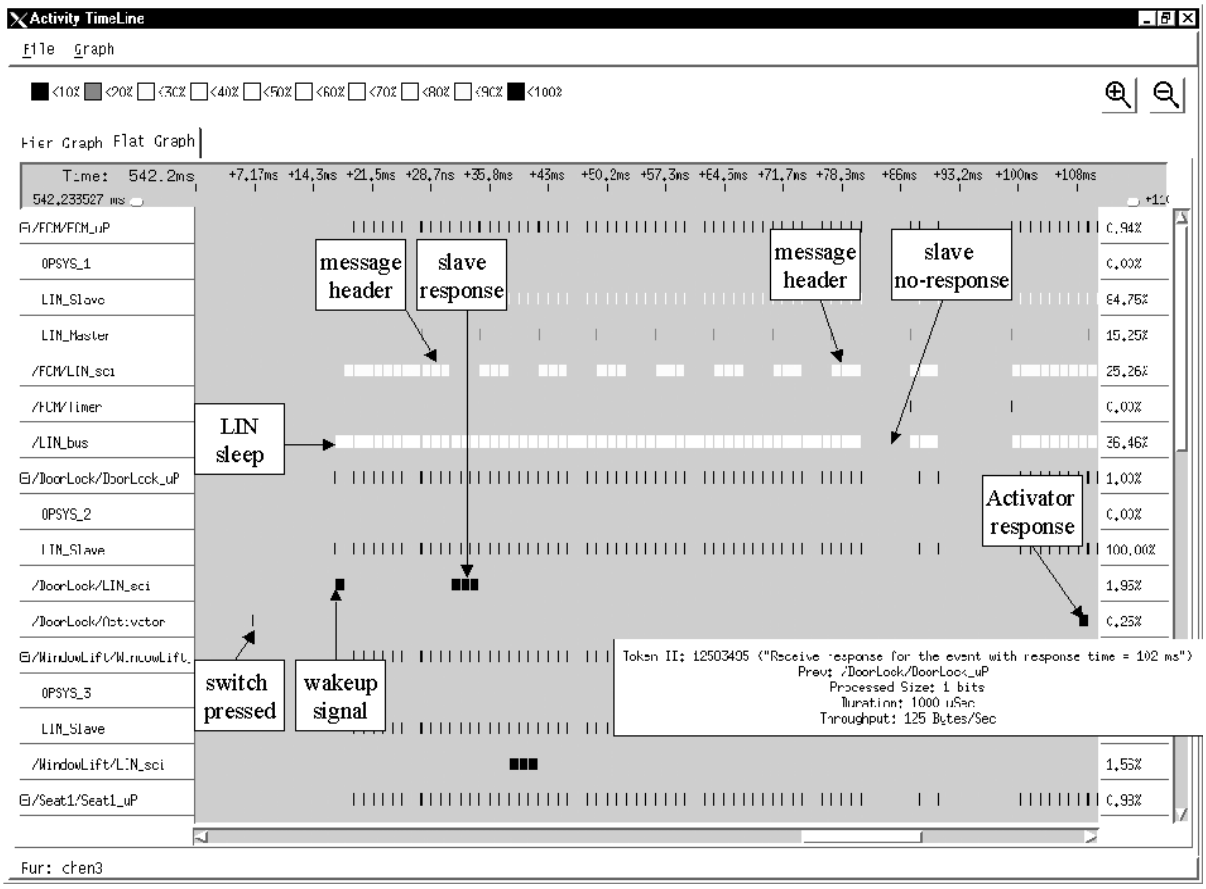


Figure 5.1 The activity chart of the LIN network simulation result

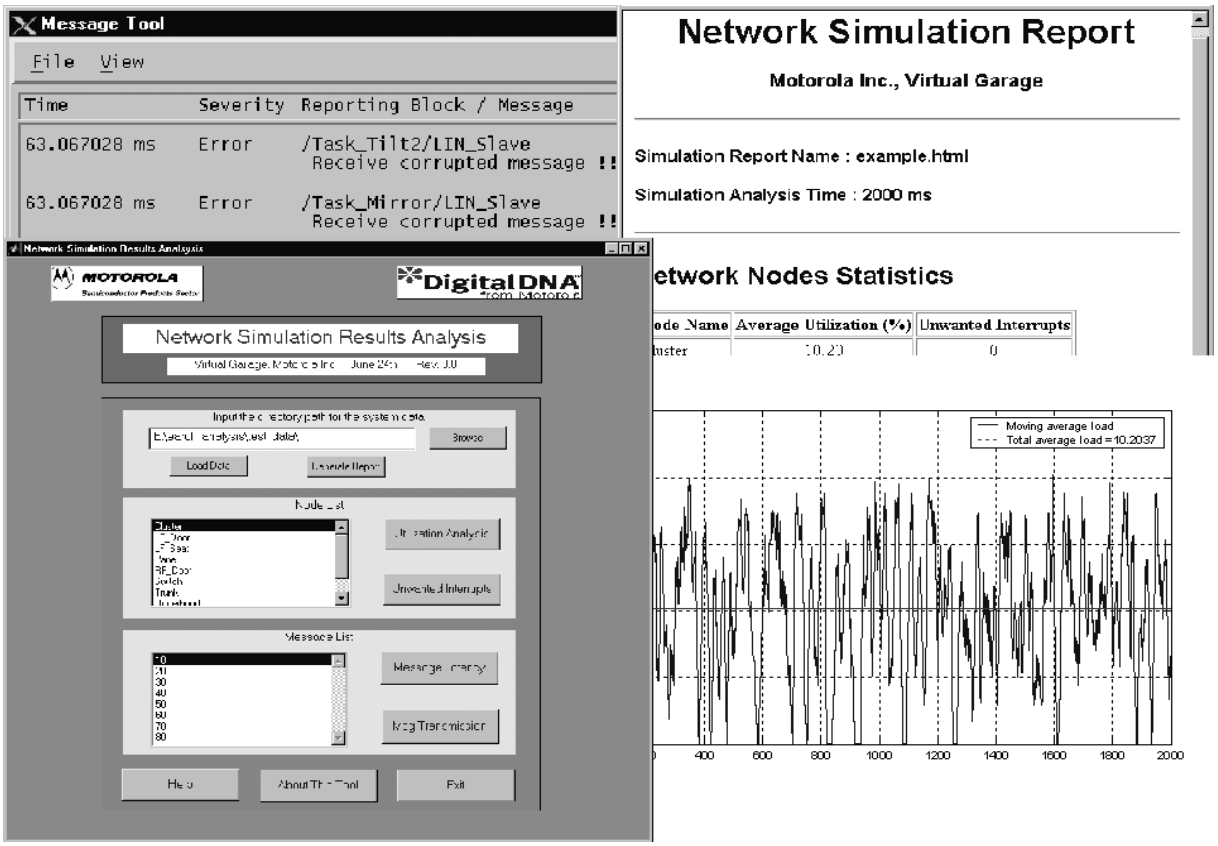


Figure 5.2 The network simulation results and analysis