ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

The Evaluation of a Performance Model for a MOST Network

Je-Hoon Lee[1], Yongrak Choi[1], Sang-Choon Kim[1], and Si-Byung Nam[1*]

[*1]Div of Electronics, Infnormation and Communication Eng., Kangwon National University, Samcheok Campus, 1 Joongang-ro, Samcheok, Gangwon-do, 245-711, Rep. Of Korea

limdg@kangwon.ac.kr

Abstract

This paper presents a MOST network model that can support the network configurations and data transmission in the network is composed with various devices employing MOST bus. It is implemented high level description language, C# and it is verified in software framework, .NET framework 4.5. The simulation results show that the proposed MOST model can capture the network configuration successfully whenever the network is initialized as well as it is changed. In addition, the proposed model can support the analysis on network payload during the data transmission. The system designers can make design decisions such as the number of nodes and the amount of data transmission using the proposed MOST model. In particular, it can be useful to evaluate the MOST devices in SoC design environment.

Keywords: MOST, automotive equipment, serial interface, bus model

Introduction

The technological developments of communication in-vehicle from traditional automotive point-to-point wiring to in-vehicle networking, using industry standards such as the LIN (local interconnect network), SAE J2602, CAN (controller area network) and FlexRay have enabled carmakers to add many new electronic features for comfort, convenience and infotainment, as well as improved control of various subsystems. Recently, most vehicles have embedded high-grade sound, CD changers, navigation systems, and an additional multimedia and communication system. Thus, it is desirable to introduce a new backbone network that can support various data transfer modes with a wide allowable bandwidth for encompassing those various embedded in-vehicle devices.

MOST (media oriented systems transport) is a bus standard for vehicle multimedia networks capable of transferring high-quality audio and video data, packet data, real-time control commands, and other signals at a maximum of 150 Mbps. MOST is a high-speed multimedia network specification that is optimized by the automotive industry by defining the physical and the data link layers [1-2]. MOST technology is used in almost every car brand worldwide. However, consumer demands for new applications that support the better-known, user-friendly, multimedia side of the internet are increasing. For example, typical car navigation simply gives the precise direction to the destination along roads using a GPS and map database. It was created to provide the fastest route when considering real-time traffic information that is obtained from the internet. In addition, the number of vehicle mounted devices should increase owing to the advent of new gadgets.

In order to evaluate automotive equipment employing a MOST interface, it is necessary to analyze the functional behaviors in the MOST network using a simulation tool or equivalent model. There is a commercial simulation tool for the MOST system implemented in SoB (system on a board) design methodology [3]. However, it is not suitable to use in a SoC (system on a chip) design environment owing to the lack of performance estimation and verification functions performed at system-level bus model. Recently, various convergent technology for traditional automotive devices have been introduced, and they should force engineers to evaluate them in the SoC design environment. Thus, it is desirable to evaluate the new simulation model for the MOST interface, in particular, suitable for a SoC design environment. This paper presents a new simulation model for the MOST standard that can provide simulation results during a MOST network configuration and data transfer phase. It will help to integrate the MOST interface into various automotive applications.

The Proposed Behavioral Modeling for MOST Network

A MOST network is able to manage up to 64 MOST devices in a ring configuration. In this network, one device is designated a controller, network master, and the others becomes slaves, that are, network slaves [4].

ISSN: 2277-9655
Impact Factor: 1.852

Each device includes external host controller having various function blocks and INIC (intelligent network interface controller) as shown in Fig. 1. In addition, each device connected the cables that compatible with a MOST specification. In particular, a function block is an object for controlling dedicated functions that used in MOST networks and it is well defined in a MOST specification. The MOST specification uses the term "function" to include both properties and methods. The properties describe or change the condition of the function to be controlled and the methods execute actions which, after a specified period of time, achieve a result [4-6]. In MOST specification, there are various function blocks for controlling applications as well as for managing the network. These function blocks also define the interface of an application to be controlled. Function block is abbreviated to FBlock in this paper. An INIC is responsible to perform the network service including network configuration and data transmissions. The proposed behavioral model is based on the modeling of behaviors of INIC and it is to emulate the status of network including network management and data transfer. It will help to verify the functional and communicated behaviors of the MOST devices.

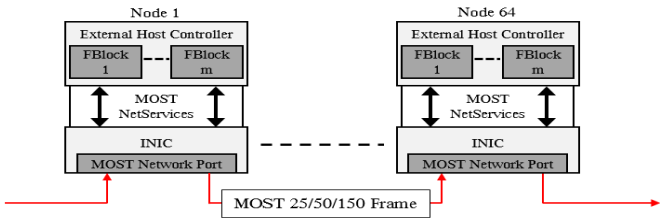


Fig. 1. MOST network architecture.

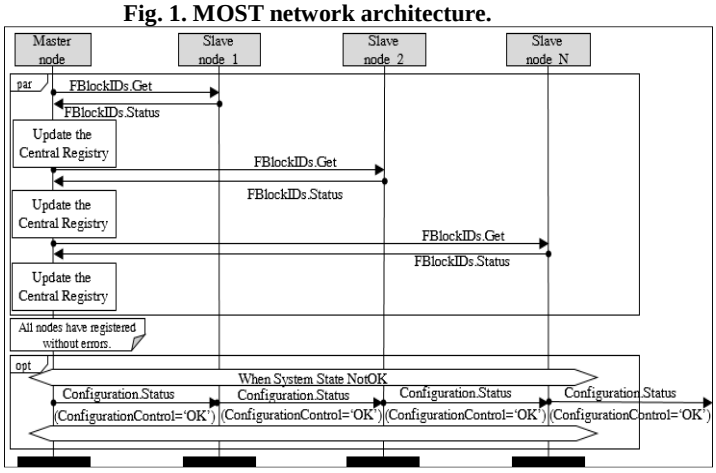


Fig. 2. Operational sequence of a system scans by the *NetworkMaster* in MOST.

Network Initialization and Re-construction in MOST network

The proposed MOST network model includes the behavioral model for network management as well as the data communications. First, we make a behavioral model for network management. This model is responsible to configure the network when the network is initialized. It is also responsible to manage the network re-configuration whenever the connection with MOST devices is changed.

In a MOST specification, there are four major function blocks for network management functions such as the NetBlock, PowerMaster, NetworkMaster, and the ConnectionMaster. A NetBlock is responsible for the administration of a device. After wake-up of a MOST network, the network master builds up all communicative relations with slaves via a system scan, that is, start-up as shown in Fig. 2(a). After successful system initialization, the communicative relations between the individual slave components are established and the current system configuration results are stored in central registry in network master. The latter lists all function blocks existing in the network, and the respective address of each device, on which a function block is implemented. All other devices in the ring are called network slaves. They may have a decentralized registry in network slave, a subset of the central registry. It comprises the function blocks of the communication partners of the device.

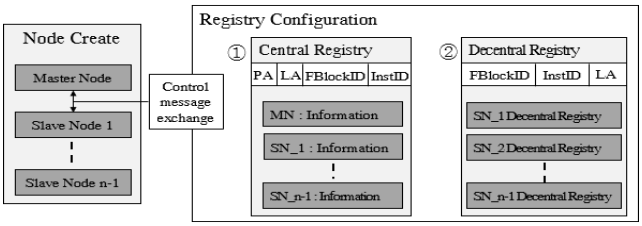
If the number of active nodes changes after start-up, an event, NCE (network change event) that indicates the change occurs. The NetworkMaster performs a system scan in which it queries the FBlocks of all devices via the function FBlockIDs to determine the system configuration after NCE occurs. Figure 2 shows the operational sequence of a system scan. After the initial system scan, devices may also inform the NetworkMaster of new or disabled FBlocks by sending a status message of the function FBlockIDs with a new list of FBlocks. NetworkMaster requests the information of slaves by sending control message, FBlockIDs.Get(). Then, the NetworkSlave returns control message, that is, F.BlockIDs.Status(FBlockDList) as shown in Fig. 2 in the par section. The obtained result is to update Central Registry. This process is repeatedly performed for all NetworkSlaves. If all nodes have registered in Central Registry without errors, initialization process is completed. The opt section describes the case that the NetworkMaster sends the Control message Configuration.Status(OK), if the network is in the NotOK state. The network configuration information obtained from all nodes is stored in Central Registry in a NetworkMaster. It contains the logical addresses and the corresponding functional blocks for all nodes that is mapped with the node position address as shown in Fig 3. This process is achieved by exchanging control message between the NetworkMaster and NetworkSlaves as shown in Fig. 4. The type of control messages and the

ISSN: 2277-9655
Impact Factor: 1.852

sequence is shown in Fig. 4. Thus, Central Registry contains all network information, while all slaves have a Decentralized Registry. In order to construct the Decentralized Registry for each NetworkSlave, a NetworkSlave requests the information to the NetworkMaster sending control message, CentralRegistry.Get(FBlockID, InstID). The NetworkMaster transfers FBlockID, InstID, and Logical address of the corresponding NetworkSlave using, CentralRegistry.Status(FBlockInfoList).

The proposed MOST bus model is commenced to initialize the MOST network as shown in Fig. 5. After startup signal is issued, the proposed model generates single network master node and the number of network slaves using the function, master_create() and slave_create(), respectively. Simultaneously, the Central Registry for network master and Decentral Registry for slaves are initialized. Then, the proposed MOST bus model starts to scan the MOST network for querying all nodes present about their function blocks. It addresses each node position address. The queried node informs the network master about its stored logical node address and the function blocks ready for communication. Whenever network master receives the information of each slaves, it updates the information stored in Central Registry. It contains the position address, logical address, FBlockID, and InstID for the corresponding slaves. After system scan is completed, the network master stores all information for every slave. Then, it copies the contents of Central Registry to Decentral Registry. And the process of MOST network initialization is completed.

If there is any change in the current network configuration, it enters NEWEXT state to update the network information. If the node is newly added to the current MOST network, the network master starts the scan process. The address of the added slave initially set to 0xFFFF and the network master assigns the address of the slave by updating the Central Registry. If the node connected is disconnected, the network master initializes the Central Registry using the function, that is, CentralRegistry_Delete() as shown in Table 1. Then, it starts to initialize the MOST network again. In the same way of network initialization process, the network master queries the information of all slaves separately. After the Central Registry is fully constructed, the network master broadcasts the contents to all slaves. Figure 5 shows the flowchart of the network initialization process of the proposed MOST model.



***MN:Master Node *PA:Position Address *SN:Slave Node *LA:Logical Address**

Fig. 3. Illustration of *Central Registry* and *Decentral one*.

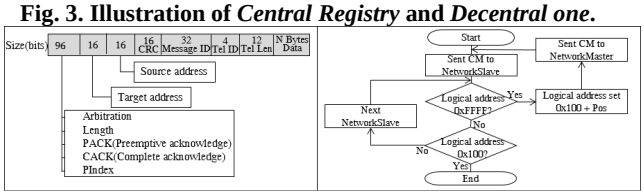


Fig. 4. (a) Control message format according to the kind of operations, (b) the flowchart to query the current status of a property.

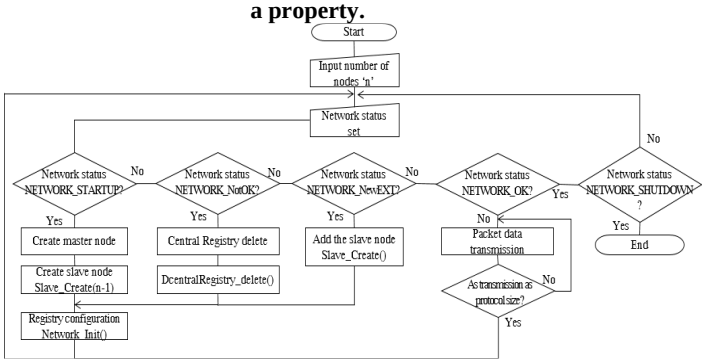


Fig. 5. MOST network model flow chart.

Packet Data Transmission via MOST Network

MOST is a multi-channel network that can support the data transport mechanisms for control, streaming and packet data. The new MOST specification Rev. 3.0 distinguishes between synchronous and isochronous data. The latter can be transported only by MOST150 specification.

The control channel provides control message service for network administration. It is secured by a CRC with automatic retry. A MOST supports three different types of control messages according to the allowable bandwidth. The control message length is varied depending on the specifics of the message. However, the maximum number of single frames is limited to prevent the control channel from occupying too much bandwidth. Second, a MOST supports streaming data channel for synchronous data and isochronous one. A synchronous data is used to transmit the real-time multimedia data such as audio and video data. It uses a application message service to establish the connection between nodes. A master node sends control message to establish node connection before the data transmission. There is no repetition in the case of

ISSN: 2277-9655
Impact Factor: 1.852

communication errors. In particular, MOST150 introduced isochronous transfer for high density multimedia data, in example, MPEG video streams. Isochronous channels are handled in much the same way as synchronous channels in MOST. Third, a MOST supports packet data channel that provides transmission of longer data packet and control data. A conventional token-ring is introduced for the arbitration to solve the contention. It uses a data link layer protocol such as the TCP/IP protocol or MHP (MOST high protocol). The node having token can transfer data packet on the packet data channel. Figure 6 shows the flowchart for packet data transmission.

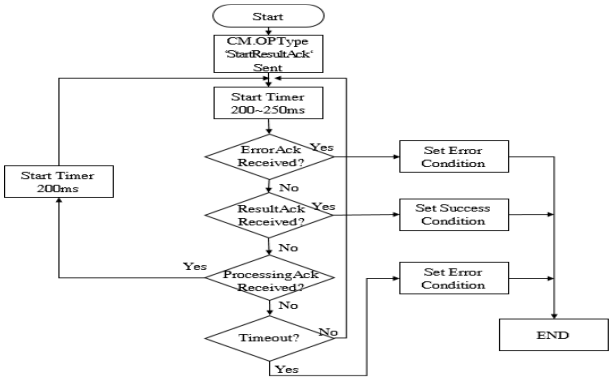


Fig. 6. Flow chart of packet data transmission.

Throughput Model for Data Transmissions

This section introduces the detailed explanations of the proposed throughput model for MOST network. First of all, MOST network employs a conventional token-ring for the arbitration to solve the contention. The node having token can transfer data packet on the packet data channel. To model a MOST network properly, we must first make assumptions about the initial probabilities to request a network ownership for each node. Then, we make formulas to calculate the probabilities to receive the grants of network ownership from an arbiter for every node separately. Next, we calculate again the probabilities to request a bus ownership for each master for next round and we calculate probabilities to receive the grant for the corresponding master. Finally, we make the general formulas for requesting bus ownerships and receiving the grants for all masters until the allotted data transmissions are completed.

The proposed throughput model adapts a probabilistic analysis method for each node in MOST network. There are four major assumptions regarding the proposed model. First, we assume the proposed model is developed in the shared bus based on token-ring architecture. It is assumed that once a node possesses the network ownership, other nodes cannot access the network until the node releases the ownership of network. Second, the arbitration policy is similar with original TDM (time division multiplexing) technique. When we assume that the proposed model is based on a MOST network consists of N nodes from M_1 to M_N, the timing wheel for time division multiplexing consists of same number of time slots that are mapped to each master. The alphabet represents the sequence of the allotted time slots. Each time slot can span single transactions via MOST network for the corresponding node. Third, we define the new term, round as the time that taken the timing wheel is rotated once. Before each round starts, the nodes request bus ownership to transfer the data. Then, an arbiter assigns the timing slots for each node. If the node does not request the network ownership, the allotted time slot is wasted. Fourth, we assume each node has an ideal buffer that is large enough to store the data shall be transferred within the time limit allotted for each bus task.

A probability to request network ownership for each node

The proposed performance model is based on a bus task model. Many SoC system applications can be seen as being object based, in example, they can be characterized by a dynamic set of objects that can dynamically appear or disappear depending on operational scenarios of target system. We consider the set of objects as a set of real-time bus task. A bus task is a basic unit for the data transmissions on an on-chip bus and it is partitioned into one or more bus tenures which are defined as the maximum time any node can hold the bus prior to relinquishing the bus to other contending nodes [9]. Each node issues the network ownership request repeatedly until it wins the contention. Thus, the number of requests that are issued by each node shall be changed during the different bus tasks are performed. That is to say, each node should have the set of different numbers of network requests during each bus task.

The bus task is periodic in nature with a fixed execution requirement each period. If multiple nodes connect to MOST network, multiple requests can be outstanding. Then, the timing wheel in token-ring arbitration matches the outstanding request and the time slot for each node. In this paper, we refer to a set of outstanding requests to request network ownership as a task set. This task set is performed in one or more rounds which are in turn partitioned as multiple time slots is the same number of the nodes.

First, we commence to generate the formulas related with the probabilities to request network ownership for each node. Let the set of nodes be M_1, M_2, … , M_n. All nodes have the allotted data transmissions according to the operational scenarios of target SoC. At the beginning of each task, let the number of the data transmissions assigned by each node be $N_D(M_1)$, $N_D(M_2)$,

… , $N_D(M_n)$. Total number of data transactions that shall be performed in the specific bus task, $T_D(i)$, is obtained from the sum of the allotted data transmissions for all nodes as $N_D(M_1)+N_D(M_2)+ \dots +N_D(M_n)$. For this analysis, deadlines are coincident with the end of period. A task transfers all allotted data from the source node into the destination one before the end of period, T_j that is the number of cycles for each master during a task. Consequently, the probability to request network ownership for each node should be significantly dependent upon the relationship between the number of the allotted data transmissions and the length of deadline during each task. These probabilities are calculated at the beginning of each task and all nodes renew their probabilities to request network ownership whenever every round is started.

We set the initial probability to request network ownership for *k*-th node at the beginning of each bus task to $P_{R1}(M_k)$. We only consider the amount of data transferred and the required number of cycles for deadlines to simplify the proposed model. The initial probability to request bus ownership for each node should be in proportional to the number of allotted data transfers. In addition, all nodes have same length of deadline in the same ask. Thus, the initial probability to request bus ownership for *k*-th master is obtained by dividing the number of assigned data transfers, $N_D(M_k)$ by the allowable number of cycles for its deadline, T_j as shown in Eq. (1).

$$\mathrm{P}_{R0}(M_k) = \frac{N_D(M_k)}{T_j} \tag{1}$$

Then, the initial probability to issue the request bus ownership should be updated at the beginning of every round. $P_{Ri}(M_k)$ represents the updated initial probabilities to request network ownership for *k*-th node at the beginning of the *i*-th round. There is a closed relationship between initial probability for each bus task, $P_{R1}(M_k)$ and the updated probabilities for the successive rounds, $P_{Ri}(N)$. The more data transfers are successfully performed in previous round, the fewer requests are pending in the present round. Consequently, the initial bus request probabilities for two consecutive rounds, $P_{Ri-1}(M_k)$ and $P_{Ri}(M_k)$, are linked to the probability to receive grants from bus arbiter in the previous round.

The probability to request bus ownership for *k*-th master, $P_{Ri}(M_k)$ at the beginning of *i*-th round can be obtained from Eq. (2).

$$\mathrm{P}_{R_i}(M_k) = \frac{R_{Di}(M_k)}{\sum_{i=1}^{n} R_{Di}(M_i)} \tag{2}$$

Here, *i* represents the number of rounds. $R_{Di}(M_k)$ represents the remaining data transfers for *k*-th node and $\sum_{i=1}^{n} R_{Di}(M_i)$ represents total sum of the remaining data transfers for all nodes.

A probability to be granted network ownership for each node

Based on the characteristics of TDM arbitration, the probabilities to receive the bus ownership for each node can be obtained as follows. We first consider the TDM-based MOST network consisting of *k* nodes and the different numbers of time slots in the timing wheel are assigned for each node. Let the number of the time slots that are assigned for each node be $N_T(M_1)$, $N_T(M_2)$, … , $N_T(M_n)$. Total number of time slots in timing wheel for each round is obtained from the sum of the allotted time slots for all nodes, $N_T(M_1)+N_T(M_2)+ \dots +N_T(M_n)$. In a first round during each task, the probability to be granted for *k*-th node, $P_{G1}(M_k)$ can be described as Eq. (3).

$$\mathrm{P}_{G1}(M_k) = \{\mathrm{P}_{R0}(M_k) \times N_D(M_k)\} \times \frac{N_T(M_k)}{\sum_{i=1}^{n} N_T(M_i)} \tag{3}$$

where, $P_{R0}(M_k)$ and $N_D(M_k)$ represents the initial probability to request network ownership and the number of the allotted data transfers in the task for *k*-th node, respectively. The number of requests for *k*-th node is given by $P_{R0}(M_k) \times N_D(M_k)$. Then, the probability to receive grants from the arbiter depends on the ratio of the number of allotted time slots for the corresponding node and total number of time slots in timing wheel. This result is obtained as the second term in Eq. (3). Consequently, the probability to receive the grants for *k*-th node in first round is obtained from Eq. (3). This process is repeatedly performed for all nodes separately to obtain the number of granted time slots for the corresponding node.

After the first round is completed, the number of pending requests will be issued by each node becomes changed according to the number of successful data transfer in first round. The number of pending requests in second round, $P_{R2}(M_k)$, is the results of the subtraction between the number of initially allotted data transfers, $N_D(M_k)$ and the number of data transfers successfully performed in the first round, $P_{G1}(M_k)$. This result is also impact on the probability to request bus ownership for *k*-th node, $P_{R2}(M_k)$ at the beginning of 2nd round as described in Eq. (4).

ISSN: 2277-9655
Impact Factor: 1.852

$$P_{R_2}(M_k) = \frac{R_{D2}(M_k)}{\sum_{i=1}^{n} R_{D2}(M_i)} \quad (4)$$

Here, $R_{D2}(M_k)$ represents the remaining data transfers for *k*-th node at the beginning of second round and it is obtained from $N_D(M_k)$-$P_{G1}(M_k)$. Similarly, total sum of the pending requests for all nodes is obtained from the sum of all remaining data transfers and it can be represented as $\sum_{i=1}^{n} R_{Di}(M_i)$.

The number of pending requests for each node is updated whenever the previous round is completed as depicted in Eq. (5).

$$R_{Di} = N_D(M_k) - \sum_{i=1}^{i-1} P_{Gi}(M_k) \quad (5)$$

Here, $N_D(M_k)$ represents the number of pending requests for *k*-th node at the beginning of each task. In addition, $\Sigma P_{Gi}(M_k)$ is the sum of data transfers that are successfully performed until the previous round. Whenever each round is completed, the node update the number of pending requests using the result of Eq. (5) and this result is used in Eq. (2) to calculate the renewed probability to request the network ownership in the next round, $P_{Ri}(M_k)$. Finally, this result is used to calculate the probability to be granted for *k*-th node in *n*-th round, $P_{Gn}(M_k)$ as shown in Eq. (6).

$$P_{Gn}(M_k) = \{P_{Rn-1}(M_k) \times R_{Dn-1}(M_k)\} \times \frac{N_T(M_k)}{\sum_{i=1}^{n} N_T(M_i)} \quad (6)$$

As shown in Eq. (6), a throughput of MOST network employing TDM arbitration policy significantly depends on the number of grants from a bus arbiter since it impacts on the number of pending requests and the probability to request network ownership every time the round is passed. In particular, the number of allotted time slots for each node should be important.

Simulation Results

The proposed MOST bus model is evaluated using .Net Framework 4.5. The MOST network comprises single network master and *n* slaves. They are connected in a ring topology. The maximum number of nodes in the ring is limited as 64. The proposed MOST model can be divided into two major blocks. The one is network management and the other is data transmission.

First, we construct MOST network model simulator with a network master and 63 slaves as shown in Fig. 7. After MOST network wake-up, the network initialization process is commenced. A network master starts a network scan to obtain the network connection information from the nodes that are connected. Each slave transfers the information that is stored in Central Registry. This process is repeatedly performed until all slave nodes complete the data transmission. Then, network master store the results in its Central Registry. After the network initialization process, the network master starts to broadcast the stored information in Central Registry. Whenever the slave node receives the network information for the corresponding node, it stores the information in its Decentral registry as shown in Fig. 7. Figure 7 shows final results of the network initialization process and all slave nodes successfully store their network information in their Decentral registry. Thus, this result shows that the proposed MOST network model successfully construct the MOST network.

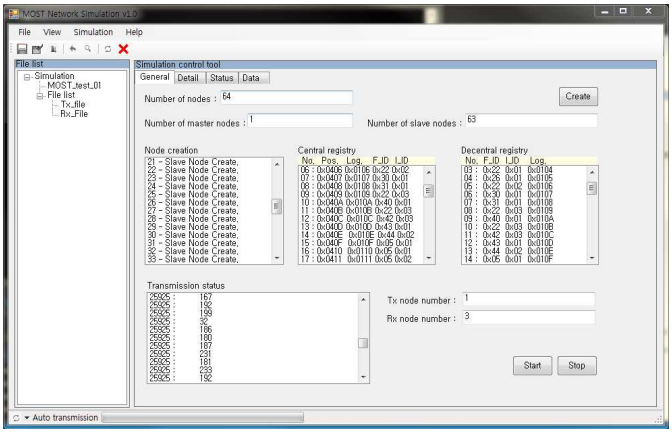


Fig. 7. Simulation results from the proposed MOST network simulator after all nodes in MOST network complete the network initialization process.

Second, we simulate the internal status in MOST network during the data transmissions. In this simulation, we change the length of packet that are allowable in MOST specifications in order to analyze the effect of the size of transmitted data packets on the throughput. Theoretically, the maximum packet length on the packet data channel in 48 bit addressing mode is 1,522 bytes. In MOST150, the length of header is 12 bytes. Thus, the maximum length for stream and packet data should be 93 quadlets, that is, 372 bytes. Thus, it requires at least 5 frames to transfer the packet data with maximum packet length. It introduces the loss of 81 quadlets in MOST protocol and it represents the device does not use the allowable bandwidth efficiently. From the simulation during the data

ISSN: 2277-9655
Impact Factor: 1.852

transmission, we obtain the length of packet data for the MOST frame without significant loss of allowable bandwidth as shown in Table 2. For example, we set the length of packet to the maximum length that are allowable in MOST specification, that is, 64 quadlets. In this case, it requires the number of frames shall be transferred are 6. On the contrary, the MOST network transfers 384 times repeatedly when we set the length of packet to the minimum length that are allowable in MOST specification, 1 quadlets. Each frame should include the header to store the information related with data transmission. Thereby, it becomes an obstacles to enhance the throughput of MOST network.

Table 1. The number of data transmissions for each node.

Data type	Unit	Value											
Packet data area	Quad.	64	48	32	14	16	12	8	6	4	3	2	1
The number of frame	EA	6	8	12	16	24	32	48	64	96	128	192	384

Finally, we show the comparison results between the proposed bus model and the typical high-level simulator, MaxSim as shown in Fig. 8. The difference is reduced with a growing number of data transactions. The difference for the first bus task is almost 19% and the slope of the difference decreases with a growing number of data transmissions. It is due to the fact that the proposed bus model is based on a probabilistic analysis method. This empirical probability of an event is an estimate that the event will happen based on how often the event occurs after running an experiment. If the number of trials becomes huge, this probability is closed to the value of theoretical probability. Consequently, the average difference between the proposed bus model and the typical high-level simulator is almost 11%.

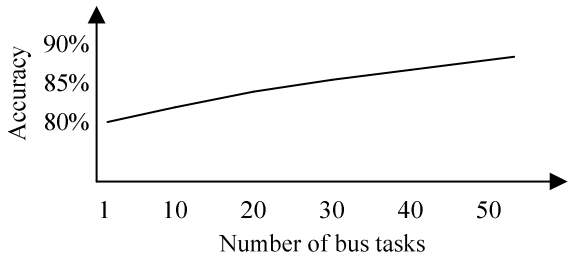


Fig. 8. Difference comparison according to the number of bus tasks

Conclusion

A MOST is a high-speed multimedia network specification that is optimized by the automotive industry. This paper proposes a MOST network model that defines the physical and the data link layers in software evaluation environment. The proposed MOST bus model is to analyze the status of network during the network configuration and data transmission. Thus, it can be used to develop the MOST applications without configuration of MOST network in practice. The proposed MOST bus model can point out the real-time changes in registries used in MOST nodes whenever the network configuration is initialized and changed. In addition, it can show the amount of data transmission on MOST bus for each node. Thus, it is possible to analyze the amount of data shall be transferred according to the transferring type without significant payload overhead and it helps to determine the number of node connected in target MOST bus. In addition, the difference between the proposed bus model and typical cycle-level simulator is under 11%. In particular, the proposed bus model does not require the cycle-accurate system model of target SoC. Consequently, the proposed bus model is useful for the early evaluation environment of target SoC. The simulation results prove the efficiency of the proposed MOST bus model. Thus, the proposed MOST bus model can be useful to evaluate the MOST interface chip in SoC design environment.

References

[1] *MOST Cooperation, http://www.mostcooperation.com.*
[2] *S. Proferl, M. Becht, P. Pauw, "150Mbit/s MOST, the next generation automotive information system," Proc. of ICTON, pp. 1-2 , (2010)*
[3] *http://www.smsc.com/Products/Automotive/MOST/.*
[4] *Andreas Grzemba, MOST:The automotive multimedia network, FRANZIS (2011)*
[5] *MOST Cooperation, MOST Specification Rev.3.0, (2010)*
[6] *MOST Cooperation, MOST Dynamic Specification 3V0-2, (2012)*
[7] *Y. Jin, R. Liu, X. He, Y. Huang, "A distributed power management design based on MOST networks," ComSIS, vol.8, no.4, pp. 1097-1115 (2011)*
[8] *A. Braun, O. Bringmann, D. Lettnin, W. Rosenstiel, "Simulation-based verification of the MOST net interface specification revision 3.0," Proc. of DATE10, pp.538-543 (2010)*
[9] *K. A. Kettler, D. I. Katcher, and J. K. Strosnider, "A modeling methodology for real-time/multimedia operating systems," Proc. of IEEE Real Time Technology and Applications Symposium, pp. 1-15, 1995.*