EVALUATION OF COMMUNICATION LATENCY IN VMEBUS-BASED REAL-TIME CONTROL SYSTEMS

Jaehyun Park and Kang G. Shin

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122, U.S.A.
{jaehyun,kgshin}@eecs.umich.edu

Abstract: Although the VMEbus has been widely used for real-time control systems, its real-time performance has not been evaluated thoroughly. This paper evaluates the latency of interprocessor communication in a VMEbus-based real-time control system. To minimize communication latency, mixed priority/round-robin (MPR) bus arbitration is proposed and its performance is evaluated along with standard PRI and RRS arbitration. The latency is evaluated via computer simulations and experimental measurements using a high-resolution timing-measuring instrument that can be directly plugged into the VMEbus.

Key Words: Real-time systems, performance evaluation, multiprocessor systems

1. INTRODUCTION

Multiprocessor systems are considered as one of the most promising architectures for high-performance real-time control applications. Essential to the multiprocessor-based real-time control system is the backplane bus to transmit and receive real-time data for input/output interfacing, servo-level control, job scheduling and machine monitoring.

With the advance of microprocessor technology, several backplane buses are increasingly used for real-time control systems. The VMEbus (Peterson, 1993) is one of the most widely-used industry standard backplane buses for high-performance control systems. Over the past decade, many VMEbus-based multiprocessor systems have been proposed and built for a wide range of real-time control applications like machine control & monitoring, and large-scale process control systems (Burnley, 1988; Laduzinsky, 1991). While the flexibility and stability of VMEbus have been thoroughly studied for these applications, its performance has not been evaluated very carefully. However, since timeliness is one of the most important properties that a real-time control system should possess, and many real-time control systems use VMEbus as their backplane bus, it is important to evaluate the communication latency of VMEbus in order to ensure the real-time operation of the underlying control system.

Using generalized stochastic Petri nets (GSPNs) is a popular approach to the analysis of multiprocessor systems with the backplane bus (Marsan *et al.*, 1982; Marsan *et al.*, 1986). However, Marsan *et al.*'s work (1982; 1986) focused on the architectural performance of multiprocessor systems, rather than the performance of the backplane bus itself. Furthermore, their models were based on the conceptual backplane bus instead of commercially-implemented backplane buses like the VMEbus. The performance of a VMEbus-based multiprocessor system is evaluated by Hassapis (Hassapis, 1993) using a high-level Petri net modeling. He modeled and analyzed the behavior of a VMEbus-based multiprocessor in order

to compute the probability of dynamic failure or missing a deadline. The main intent of this paper is to evaluate the latency of VMEbus while considering the real hardware parameters used in commercial products.

The data transferred over the VMEbus in a real-time control system can be classified into three classes according to their timing constraints: (1) hard real-time, (2) soft real-time, and (3) non real-time. Hard real-time data represent such critical control information as an emergency stop signal and obstacle-avoidance commands, and missing their delivery deadlines may cause serious safety or financial problems. Soft real-time data are desirable to be transferred in a certain time limit, but missing their deadlines does not cause any serious problem to the system (it may cause some inconvenience). Although non real-time data, also called *best-effort* data, do not have any timing constraint to meet, it is still desirable to transfer them as fast as possible. In a VMEbus-based system, because there is no separate backplane bus for each class of data, the timing constraints should be satisfied by scheduling the use of the backplane bus, which is determined by bus arbitration. Even though seven levels of VMEbus interrupts are defined for transferring data based on their urgency, the interrupt handler for hard real-time data still needs to acquire bus mastership in order to send the data.

In this paper, the communication latency of VMEbus is evaluated using computer simulations and experimental measurements on a real machine. In Section 2, the components of VMEbus latency are identified and its least upper bound is estimated. To evaluate the VMEbus latency, the VMEbus operation is modeled with a GSPN. Using the GSPN model, the VMEbus latency is evaluated while varying the traffic and arbitration strategy of the bus. The simulation results are presented in Section 3. Section 4 presents and compares experimental results with simulation results.

2. ANALYSIS OF VMEbus LATENCY

2.1. Description of VMEbus latency

There are two types of modules in a VMEbus-based multiprocessor system: a master module which can initiate a data transfer cycle, and a slave module which supplies and receives data to/from the master that initiated a data transfer cycle. The communication latency of VMEbus, t_{vme}, can be defined as the time from the instant that a *requester*, one of the masters, initiates a data transfer cycle to the instant that a master completely latches the data as shown in Figure 1.

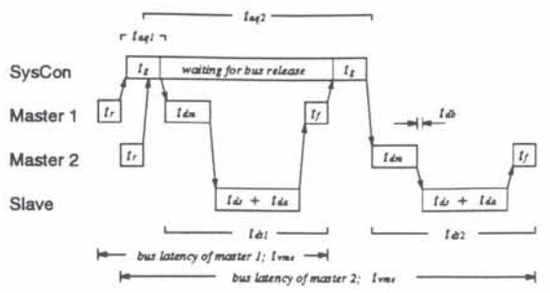

Fig. 1. VMEbus timing diagram

t_{vme} can be thought of as the sum of bus acquisition time, t_{aq}, and data transfer time, t_{dt}. The data transfer time, t_{dt}, consists of the local bus delay of master module, t_{dm}, the local bus delay of slave module, t_{ds}, VMEbus signal delay, t_{db}, and the data access time of slave module, t_{da}.

The bus acquisition time, t_{aq}, is the time required for a requester to acquire the right to use VMEbus exclusively. When the system controller detects a bus request signal (BRQ) from a requester, it grants the bus mastership to the requester unless the bus is being used by another master module. However, if the VMEbus is being used by another master module or several master modules request the bus mastership at the same time, the system controller should select one requester based on its pre-defined arbitration strategy.

The VMEbus standard defines three types of arbitration: priority-based (PRI), round-robin selection (RRS), and single level (SGL). However, because the daisy chained bus-grant signal is used for SGL arbitration, SGL can be regarded as a special case of PRI. In addition to these standard arbitration strategies, the IEEE standard (ANSI/IEEE STD1014-1987) permits user-defined arbitration (VMEbus International Trade Association, 1987).

Besides the underlying bus arbitration strategy, t_{aq} is also influenced by the bus release policy that determines how to release VMEbus mastership. There are again three types of bus release policies: release when done (RWD), release on request (ROR), and release on clear (ROC). Although the bus release policy has influence on the communication latency of VMEbus, its effect is minor compared to the bus arbitration strategy. Hence, the effect of bus release policy is ignored and only the RWD policy is assumed.

2.2. Arbitration for real-time systems

In the previous section, t_{vme} is defined as the sum of t_{dt} and t_{aq}, and t_{aq} depends on how the use of the bus is scheduled. Although PRI arbitration is commonly used for general-purpose VMEbus-based systems, because a low-priority master mod-

ule has a longer bus latency than a higher-priority master module, PR arbitration is not applicable to real-time control systems. While RRS can provide fair opportunities to all masters, its main drawback is that it cannot deal with hard real-time data. Hence, for a real-time system with the three classes of data mentioned earlier, a special arbitration strategy is necessary.

In this paper, a mixed priority/round-robin (MPR) strategy is proposed for real-time control systems. The basic idea of MPR strategy is to reserve the highest bus request level for hard real-time data and use three remain bus request levels for normal (soft real-time and best-effort) data. In MPR, while the highest-priority level has priority over other request levels, the other three request levels are used as RRS. This strategy can provide fair scheduling for all the requesters, while maintaining a high-speed channel for hard real-time data. In addition to MPR arbitration, to schedule the mixed bus traffic of soft real-time and non real-time data, ROC is used for non real-time data. Thus, the transfer cycle of non real-time data may be preempted by a real-time data transfer cycle.

2.3. VMEbus latency bound

From Figure 1, the VMEbus latency is expressed as $t_{aq} + t_{dt}$. If the local bus arbitration of a slave module is fair, t_{dt} will have a constant upper bound, $t_{dm}+t_{ds}+2t_{db}+2t_{dw}$, regardless of the bus arbitration strategy used, where t_{dw} is the memory access time. By contrast, t_{aq} strongly depends on the arbitration strategy. With PRI arbitration, t_{aq} is

$$t_{aq} \leq \begin{cases} t_{dt} + t_g + 2t_{db} & p = 1 \\ (p-1)(t_{dt} + t_g + 2t_{db}) & p > 2 \end{cases}$$

where p is the requester's relative priority with the highest priority being 1. However, this equation can be used only when VMEbus traffic is low. If VMEbus traffic is high, like

$$\frac{1}{\lambda} \leq 3(t_{dt} + t_g + 2t_{db}),$$

where λ is the bus-request rate of each requester, the bus latency of lower-priority ($p > 2$) requesters diverges. This means that PRI arbitration cannot be used for real-time control systems when bus traffic is high.

The t_{aq} of RRS is bounded by $(n-1)(t_{dt} + t_g + 2t_{db})$, where n is the number of master modules. Although RRS arbitration provide the same communication latency for all requesters, it has no provisions for handling hard real-time data communications.

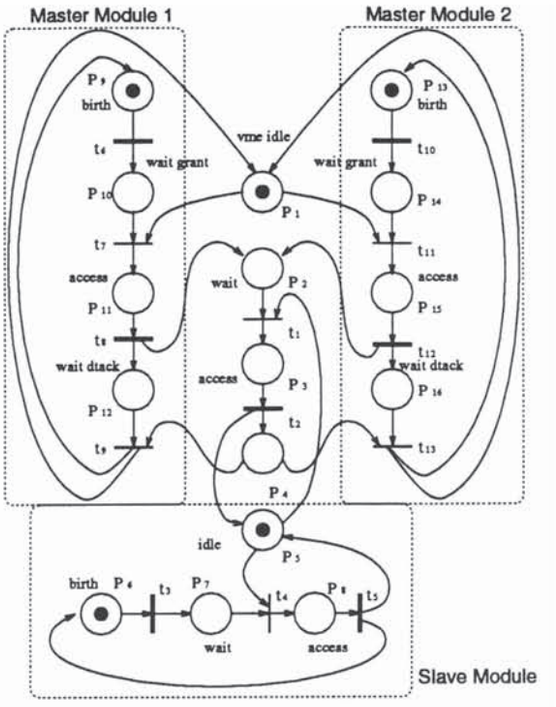

Fig. 2. The GSPN model of a VMEbus-based controller

In MPR arbitration, the bus acquisition time for normal data and hard real-time data, t_{aqn} and t_{aqh}, respectively, are

$$\begin{aligned} t_{aqn} &\leq n(t_{dt} + t_g + 2t_{db}) \\ t_{aqh} &\leq t_{dt} + t_g + 2t_{db}. \end{aligned}$$

This means that the latency of normal data communication is slightly longer than that of RRS because hard real-time data communications are usually rarer, and their latency is the same as the latency of the PRI's highest-priority level.

3. COMPUTER SIMULATIONS

To evaluate the VMEbus latency, its hardware-level operation is modeled using a generalized stochastic Petri net (GSPN), a popular tool for performance modeling (Caselli *et al.*, 1992; Molloy, 1982). Using the GSPN, one can easily model the concurrent, asynchronous event-driven, and priority-based operation of VMEbus.

Figure 2 shows a simplified example of VMEbus-based real-time control system, which consists of a system controller, two master modules, and one slave module. This model uses PRI arbitration, represented by P_1, and the RWD bus release policy, represented by P_4. Other bus arbitration strategies such as RRS and MPR can be modeled similarly by modifying P_1, and other bus release policies can be modeled by changing P_4. Transition t_6 and t_{10} represent the bus-request rate of

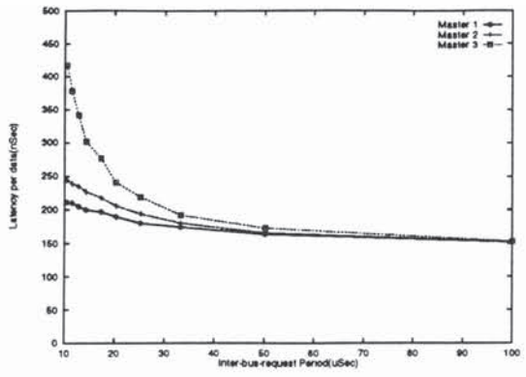

Fig. 3. VMEbus latency under PRI arbitration

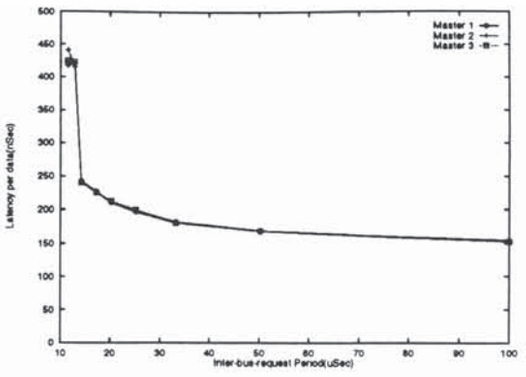

Fig. 4. VMEbus latency under RRS arbitration

each master module, which depends on the workload of each master module. t_2 determines the block size of each data transfer. If a slave module is intelligent, the local bus can be shared by a local CPU and a master module. The local bus arbitration is controlled by P_5. To model a non-intelligent slave module, the initial marking of P_6 is set to zero. From this model, the feasibility of VMEbus operation can be studied and its performance can be analyzed using the Markov chain solution (Molloy, 1982).

While this simplified model shows the general operation of a VMEbus-based multiprocessor system, more detailed modeling is required to evaluate hardware-level latency. For example, t_7 in Figure 2 represents t_{aq}. However, in the hardware level, t_{aq} is the sum of t_{dm}, t_{ds}, t_{db}, and the delay caused by VMEbus contention. To reflect the effects of these parameters correctly, a more detailed GSPN model, called UltraSan, is used for simulation (Couvillion *et al.*, 1991; Performability Modeling Research Laboratory, 1993).

The parameters used for computer simulation are derived from the VIC-068 VME controller chip (VTC Incorporated, 1990). The simulated real-time system has three master modules and one intelligent slave module. Since the number of slave modules does not affect the performance of VMEbus, one slave module can be assumed without loss of generality. The controlled plant connected to the slave module is a real-time system with 5 sensors (e.g., a 5-axis machine controller). Because the typical sampling rate of each sensor ranges from 4KHz to 15KHz, the VMEbus request rate should be in the range of 10KHz to 100KHz. At each bus operation, the block size is assumed to range from 4 bytes to 256 bytes.

Figures 3, 4 and 5 show the simulation results of the average bus latency of PRI (including SGL), RRS, and MPR arbitration, respectively, when 64 bytes are transferred in a single bus operation. According to the simulation results, if VMEbus traffic is low, the inter-bus-request time is larger than

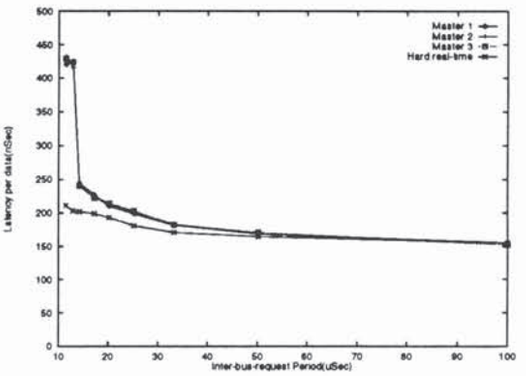

Fig. 5. VMEbus latency under MPR arbitration

$60\mu sec$, all arbitration strategies show an even bus latency for all requesters. However, under high VMEbus traffic, there is a large variation in bus latency with PRI arbitration. Figure 5 shows that using MPR hard real-time data can be transferred with less VMEbus latency.

4. EXPERIMENTS

To validate the simulation results, the VMEbus latency is measured experimentally using a fine-grain instrument, called the *VME Stopwatch (VSW)*, which has been developed for in-plug timing measurement of VMEbus operation. The VSW was implemented using a Lattice FPGA (isplsi 1048), which contains the VMEbus controller, a local state machine, and a high-speed counter. The block diagram of VSW is given in Figure 6. The basic operation of the VSW is to record events along with high-resolution (25 nsec) timestamps, which may be generated by master modules on the VMEbus.

The test environment consists of three master modules (Ironics IV3207 modules) which use VIC-068 as the VME controller and one slave module. When a master module generates an event to VSW periodically, the timestamps recorded by VSW have variations indicating the extra latency caused by bus contention.

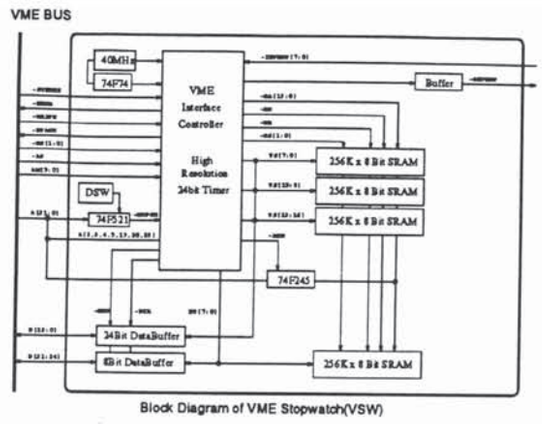

Fig. 6. A functional block diagram of VME-Stopwatch

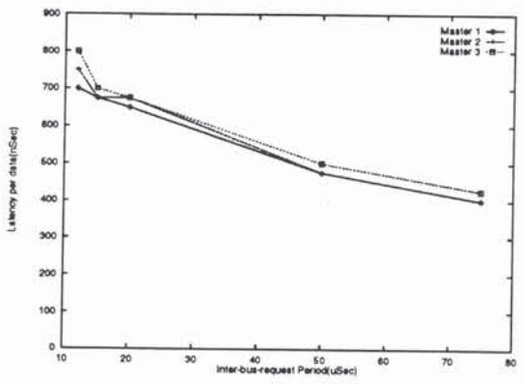

Fig. 7. VMEbus latency measurement results

Figure 7 shows the experimental measurements while varying the workload of each master module under PRI arbitration. These results show a large latency when the bus-request interval is less than $20\mu sec$.

5. CONCLUSION

In this paper, the communication latency of VMEbus for a real-time control system is evaluated. Since the bus scheduling, controlled by an arbitration strategy, has a great influence on the bus latency, a user-defined arbitration strategy (MPR), is proposed for real-time applications, and its latency is evaluated along with standard arbitration methods, such as PRI and RRS. Computer simulation and real experiments are used to evaluate the bus latency. To measure the bus latency accurately in each experiment, a high-resolution timing-measuring instrument that can be directly plugged into the VMEbus, has been developed. The evaluation results in this paper can be used for real-time scheduling and task allocation for controllers built with the VMEbus.

ACKNOWLEDGMENT

The work reported in this paper was supported in part by the National Science Foundation under Grants DDM-9313222 and MIP-9203895. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the NSF.

REFERENCES

Burnley, P. (1988). CPU architecture for realtime VME systems. *Microprocessors and Microsystems* **12**(2), 153–158.

Caselli, S., G. Conte and U. Malavolta (1992). Topology and process interaction in concurent architectures: A GSPN moldeling approach. *Journal of Parallel and Distributed Computing* **15**(3), 270–281.

Couvillion, Joseph A., Roberto Freire, Ron Johnson, W. Douglas Boal II, Akber Qureshi, Manish Rai, William H. Sanders and Janet E. Tvedt (1991). Performability modeling with UltraSAN. *IEEE Software* **8**(5), 69–80.

Hassapis, George (1993). High level Petri net modelling and analysis of VME-based multiprocessors. *Microprocessing and Muicroprogramming* **36**, 195–204.

Laduzinsky, Alan J. (1991). An open architecture, VMEbus PLC. *Control Engineering.*

Marsan, M. A., G. Balbo and G. Conte (1982). Comparative performance analysis of single bus multiprocessor architectures. *IEEE Transactions on Computers* **C-31**(12), 1179–1191.

Marsan, M. A., G. Balbo and G. Conte (1986). *Performance models of multiprocessor systems.* MIT Press. Cambridge, MA.

Molloy, M. K. (1982). Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers* **C-31**(9), 913–917.

Performability Modeling Research Laboratory (1993). *UltraSAN Version 1.2.0 User's Manual.* University of Arizona. AZ.

Peterson, Wade D. (1993). *The VMEbus Handbook.* 3rd ed.. VMEbus International Trade Association. Scottsdale, AZ.

VMEbus International Trade Association (1987). *The VMEbus specification.* VMEbus International Trade Association. Scottsdale, AZ.

VTC Incorporated (1990). *VIC068 VMEbus interface controller specification.* VTC Incorporated. Bloomington, MN.