SIMULATION OF MULTIPROCESSOR BUS SYSTEMS FOR REAL-TIME APPLICATIONS

Andrew Kornecki
Dept. of Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, FL 32114
korn@db.erau.edu

Janusz Zalewski
Dept. ECE
University of Central Florida
Orlando, FL 32816-2450
jza@ece.engr.ucf.edu

Abstract. This paper discusses the principles of bus systems simulation for real-time applications. It uses a typical queuing model of a single bus with multiple processors, memories and I/O devices. The major criterion used in evaluation of real-time response is the bus access latency. An illustrative example of a VMEbus system with two arbitration protocols, round-robin and priority-based, for typical data acquisition workloads, is presented to document the results.

1 Introduction

Most of the modern computer systems used in real-time applications share a common bus architecture. There is a variety of popular bus standards: ISA, VMEbus, Multibus I & II, PCI, Futurebus+, to name a few [14]. The processing agents, memory, and input/output controllers share the bus access for data exchange.

For real-time systems, the critical issue is the temporal determinism of system response. The timing of computations in the system depends not only on the instruction execution time and memory access, but also on the timing of data exchange between various system components. The latter, in addition to the hardware characteristics, is a function of the specific bus architecture and the bus access and arbitration protocols. In most cases, the system has a dedicated bus controller/arbiter – a device designated to manage the bus operation and execute specific bus protocol.

The present study addresses the issue of system performance related to the services requested by agents sharing a common memory on a single bus, with respect to the bus access protocol. The objective is to develop an object-oriented simulation model for a standard bus architecture and apply this model to a case study evaluating the performance of a popular VMEbus system [2] – a leading platform for industrial real-time market.

2 Bus System Model

Bus architectures can be modeled as a version of a queueing system with customers, servers, and resources [1]. Once an appropriate model is developed the system can be evaluated either via analytical study, using theoretical underpinning of queueing theory, or by performing a simulation study and running exhaustive computer simulation experiments. The analysis route is often faster but it suffers from serious limitations. Analytical models are built under very strong simplifications, mostly dealing with *ideal* distributions, steady state conditions, etc. The simulation models are more flexible, but require lengthy development and need to be carefuly and methodically verified and validated.

There are several studies in literature modeling the bus systems in such a way [6, 7, 9, 10, 11, 13]. The models, however, focus either on distributed systems and describe networking applications or discuss multiple bus architectures with only limited relevance to real-time issues.

When building the bus architecture simulation model, we focus on the bus performance. The three basic metrics, identifying the performance of any bus system, are:

- system throughput - how much data can be transmitted per time unit,
- bus utilization - what percentage of time the bus is busy,
- bus access latency (a part of bus response time) - the time needed by the bus to grant ownership to the requesting agent.

A high average system throughput is critical for soft real-time systems in various data acquisition/data fusion applications. Related high utilization of the bus indicates that the margin for system upgrade is limited. For hard real-time systems, more critical is the determinism of system response. Therefore, in a real-time study, one has to focus not only on an average response time, but need closely watch the variance and the maximal values.

The above mentioned criteria of bus performance are determined by the following factors:

- number of active agents (processors) attached to the bus,
- latency of the memory,
- bus access protocol and the arbitration method used,
- physical characteristics of the bus,
- workload characteristics of the processor operation.

Selection of the workload, including the frequency of requests and the size of packets sent over the bus (often related to the local processor cache hit/miss ratio), is critical to the performance analysis study. The workload depends on the application and must be clearly identified before simulation model is developed. As for real-time systems, we focus on the worst case scenario to determine whether or not the system timing requirements can be met.

2.1 Bus Characteristics

In this study, we concentrate on VMEbus standard. The VMEbus architecture conforms to the ANSI/IEEE Std 1014-1987 [2] specification for high-performance backplane buses with either single or multiple processors. It is a global parallel interconnect incorporating the following features:

- support for 8-, 16-, 24-, or 32-bit data transfers (64-bit in VME extension),
- asynchronous and dynamically adaptive bus protocol,
- support of up to 20 agents on its backplane,
- three different arbitration algorithms.

The VMEbus signals are diveded into four functional groups:

- data transfer
- arbitration
- interrupt
- utility

A non-multiplexed asynchronous protocol allows transfers between master and slave agents. Only one transaction can be performed on the bus at any time. As the VMEbus solution is designed for multiprocessor support, three issues are considered [4, 12]:

- priority-based access by multiple masters,
- reconcilliation of different data/address width,
- organization of interrupt handling and resource partitioning.

The controller located in the first slot of the VMEbus enclosure acts as the bus arbiter between competing masters vying for bus access. A master (an active agent) initiates and controls any data transaction accross the bus. The slave (passive agent) can only respond to master requests. A single processor board may house both passive and active interfaces.

The VMEbus standard offers a high-speed asynchronous parallel data transfer. After the bus master initiates data transfer cycle, it has to wait for the slave to respond before finishing the cycle. The asynchronous operation allows the slave module to take all the time it needs to respond. In addition, no module needs to have any explicit knowledge about the timing requirements of other modules because the speed of the data transfer dynamically adapts to the speed of the two communicating modules.

The arbitration subsystem of the VMEbus has two main responsibilities:

- prevents the simultaneous use of the bus by two masters,
- schedules requests from multiple masters.

The VMEbus controller uses three different algorithms for arbitration: priority, round-robin, and daisy-chain (single-level) arbitration. The standard allows for use of other arbitration algorithms.

The prioritized arbitration technique assigns the bus to a requester with highest priority. Once the bus is assigned to the requester, the next assignment is granted to the requester with the next highest priority. Round-robin arbitration assigns the bus on a rotating basis (no priority mechanism is used). In the single-level protocol, the arbitration responds only to the highest priority line and depends on the requester's bus grant daisy-chain to arbitrate the requests.

2.2 Simulation Model Description

In a typical single-bus multiprocessor system the processors share information through shared memory modules or exchange information among themselves. One may expect more or less bus access conflicts requiring arbitration. In this view, the queuing model of the bus system includes several easy to identify objects: processors, caches or private memory for each processor, a global shared memory, and a bus.

In the present study, the simplifying assumption is to focus on interactions between the active agents (the processors requesting the bus access) and the bus. Once the bus request is made, the arbitration protocol determines whether or not the access can be granted. If the access is not granted, the requesting agent is waiting. If the access is granted, the transmission commences for the packet size not exceeding 256 bytes [2]. If the packet is larger, the requesting processor must go again through the arbitration process with the remainder of the packet. Any arbitration requires a specified constant time, also the actual transmission time of one byte (word, doubleword) is a constant function of physical bus characteristics. In a round robin protocol, the multiple requests are handled in a circular fashion in attempt to assure fair bus access. For the priority protocol, the processor with the lower slot number has the higher priority.

The probability of requesting the bus is based on the frequency of processor bus requests. The situation that a processor requests the bus may either represent the "miss" (the processor needs to fetch instruction sequence from the global memory) or the fact that the processor writes the data to or reads data from memory or I/O in a data acquisition application. In this study, we consider the scenarios applicable to the data-acquisition type of application, where individual processors are handling I/O interfaces and move packets of data to memory over the bus.

The resulting queuing model is a closed queuing network with one server (the needed resource) and many customers. The customers in this system obey the handshaking protocol used by the bus. The random distribution and parameters are representing typical workloads (request frequency, data size and timing) determined from actual data acquisition experiments. These simulation run-time values matching the assumed distribution and parameters are generated using validated random number generators.

In an asynchronous packet-switched protocol, both the memory module and the CPUs may request the use of the bus, therefore, both can be agents on the bus. In contrast, a synchronous, circuit-switched protocol does not allow the memory to be an agent and request bus services, because in this case the memory module is only a resource object and is not allowed to become the bus master. Each processor has a random number of processing time units between the consecutive memory requests. The mean memory access time must be adjusted adding the latency required to read requests when data are transfered using circuit-switched protocol.

3 Computer Model

From the perspective of evaluating the system performance, we indentified read and write services provided by the bus to the active system agents [5]. The active agents make bus requests. The bus arbitration subsystem is responsible for controlling and granting the bus resources to the agents in a timely and organized manner. Once an agent has control of the bus it performs the transmission. The direction of transmissions is irrelevant in our simplified model. The read and write services spend time on sending packets of data between the agents on the bus. The size of data to be transmitted is the function of the workload. For the purpose of this study we identified a probability table assigning each data size a certain probability. An example of a data-intensive workload is as follows:

- 20% of requests 4 bytes of data (uniformly distributed)
- 20% of requests 4-100 bytes of data (uniformly distributed)
- 20% of requests 0.1-1 kbytes of data (uniformly distributed)
- 40% of requests 1-10 kbytes of data (uniformly distributed)

The percentage and sizes can be changed depending on the application under the study.

The introduced model assumes the following :

- each processor in the system has identical clock rate,
- each processor has identical bus request frequency,
- packets on VMEbus can be divided into smaller subpackets (not exceeding 256 bytes, determined by the VMEbus hardware characteristics)
- transmission of the data packet in progress is not interrupted,
- memory access time is assumed constant,
- a processor can not be granted the bus if another request is presently waiting to be serviced by the bus (unless the priority protocol is used),
- errors due to invalid data or time-outs are neglible to the overall performance of the system,
- the data path for VMEbus is fixed at 32-bits (4 bytes) for all simulation experiments.

The required set of model input parameters can be separated into two categories, the system parameters and the workload parameters, as below:

- System Parameters
  - Number of processors in the system,
  - Bus transmission speed (time to send unit of data through the bus),
  - Arbitration time (the time required to select the next bus owner),
  - Transmission unit (the maximum amount of data transferred in a single transfer operation; read/write of a byte, word, or double word - 256 bytes for VMEbus).
- Workload Parameters
  - Interarrival time, an average time between two consecutive bus requests from the same processor (generated using a negative exponential distribution)
  - Packet size, the number of bytes to be sent in the request generated using probability table and uniform distribution
  - Protocol (priority-based or round-robin).

An object-oriented approach facilitated the design and implementation of this simplified VMEbus model. The model includes the following objects: CPU, Bus, and Report (see Appendix I). There are multiple instances of CPU objects. The role of CPU is to run its operation including appropriate genaration of requests and communication. A single instance of the Bus object is a passive resource which can be given when the request is made, and taken back when the communication of the chunk of data is complete. A single instance of the Report object keeps record of the simulation. Each object has a set of attributes to support its operation.

The model was implemented in MODSIM [8]. This popular simulation language supports object-oriented discrete simulation. In particular, the MODSIM system provides a wide variety of ready to use objects in many application libraries supporting event handling, queue and resource management, random number generation, statistical analysis and data presentation, graphic interface with animated output, etc. The inheritance feature of the system allows the developer to create new objects inheriting properties of the existion objects. In addition, MODSIM provides various methods (ASK, TELL, WAITFOR) supporting concurrency and allowing for passing simulation time. In this study, the Bus object was implemented as a descendant of the MODSIM resource object. Each object has a distinctive set of attributes representing the object state, and methods – representing object behavior.

4 Simulation Experiments

The goal of this simulation performance evaluation is to determine the behavior of the multiprocessor system under the given set of assumptions and compute the performance metrics for a set of input parameters. Three different performance measures were computed during simulation: the bus throughput, bus utilization, and access time of the bus. The factorial design was used to determine the variation and the confidence of the factors that affect performance of the bus.

We selected three potential workloads, high, medium, and low, each with request rate of 500 per second. The graphs below represent the bus utilization and bus latency (per arbitration), for two different arbitration protocols: round-robin and priority-based. It is worth noting that the theoretical bus throughput is 40 MB/sec. Utilization represents percentage of time the bus was busy transmitting data. The bus latency represents the time spent in waiting for the bus access (per arbitration). Additionaly we computed maximal latencies for each

processor. We ran ten replications for each set of parameters. The length of simulation run was arbitrary 0.5 sec. A sample output of a single simulation run produces the output presented in Appendix II. Upon request, the program can provide a detailed trace.

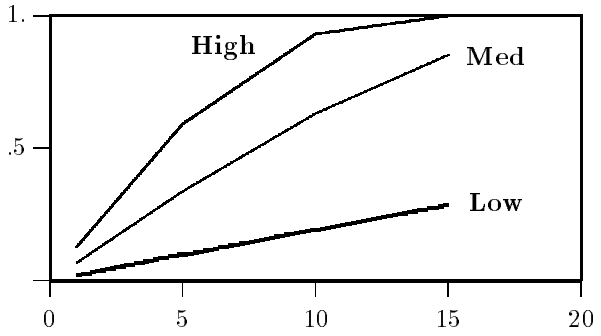

Fig. 1. Bus Utilization vs. Number of Processors for Priority-Based Arbitration (High, Medium, and Low Workloads).

Figure 1 presents bus utilization versus the number of processors for three different workloads (high, medium, and low), for priority-based arbitration. The utilization data for round-robin arbitration are almost identical and not presented here. This graph validates the simulation, since the results are similar to those obtained earlier [3]. In addition, if one wants to provide sufficient margin for real-time operation, it is clear from this graph that no more than 3-4 VMEbus processors can be used simultaneously under this workload.

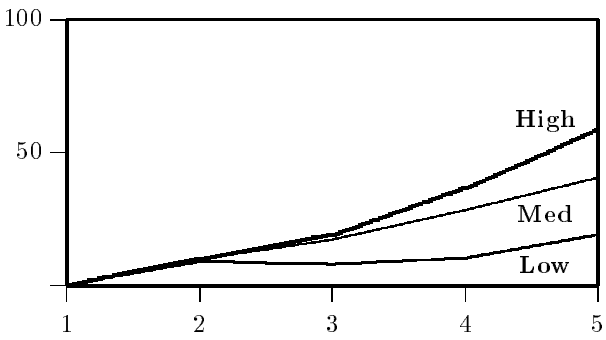

Fig. 2. Bus Access Latency (milliseconds) vs. Number of Processors for Priority-Based Arbitration and Different Workloads.

Figures 2-4 present calculations of the bus access latency time for all three workloads (high, medium, and low) and two arbitration methods: round-robin and priority-based. We used data for up to 5 processors, since from Figure 1 it is clear that using more processors, at least at a high workload, would be inappropriate for a real-time application. Figure 2 shows that for high workloads, the bus access latency almost doubles per each processor added and may significantly contribute to the overall data latency for three or more processors.

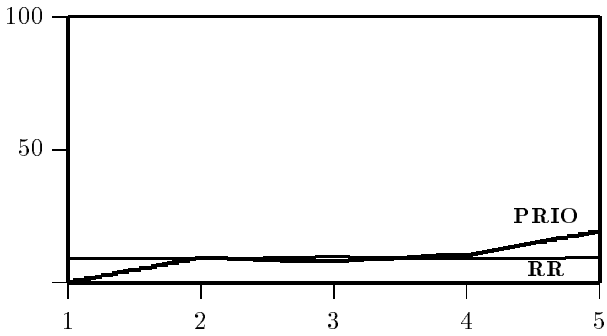

Fig. 3. Comparison of Bus Access Latency (milliseconds) vs. Number of Processors for Priority-Based (PRIO) and Round-Robin (RR) Arbitration Schemes for Low Workload.

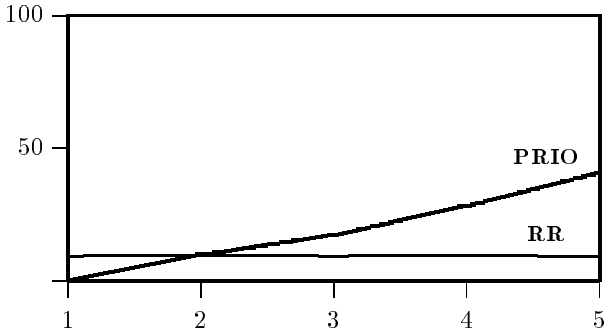

Fig. 4. Comparison of Bus Access Latency (milliseconds) vs. Number of Processors for Priority-Based (PRIO) and Round-Robin (RR) Arbitration Schemes for Medium Workload.

Figures 3 and 4 show that bus access latency remains steady for round-robin arbitration. However, when priority-based arbitration is used, the bus access latency drops significantly for the highest pri-

ority processor and may grow much above the level typical for round-robin arbitration, for all other processors, if higher workloads are used. Graphs in Figures 3 and 4 confirm that in priority-based arbitration, the lowest bus access latency is guaranteed for the highest priority processor and is at least an order of magnitude better than in round-robin arbitration. This informs the designers of real-time data acquisition systems to what extent they can rely on the use of proper arbitration method to meet respective deadlines.

5 Conclusion

The simulation results confirm previous results [3] that there is not much difference in either bus utilization or system throughput for round-robin or priority-based protocols. There is a difference, however, in bus latency. In particular, for the priority-based protocol the maximum latency is, on average, greater than for the round-robin protocol. This is due to the fact that higher-priority tasks run more frequently, at the cost of lower-priority tasks, which are delayed. However, the maximum latency of higher priority tasks is significantly shortened, which is desirable since then important tasks can run faster. In effect, the use of priority-beased protocol is especially recommended for the class of real-time applications, which have critical highest priority tasks, such as data acquisition tasks. In particular, assigning a hard real-time task to the highest-priority processor assures that the bus operation will be completed in a time bounded by the sum of arbitration time and time of packet transmission.

References

[1] Ajmone Marsan M., G. Balbo, G. Conte, Comparative Performance Analysis of a Single Bus Multiprocessor Architectures, IEEE Trans. Computers, Vol. 31, No. 12, pp. 1179-1191, December 1982

[2] ANSI/IEEE Std 1014-1987 – Versatile Backplane Bus: VMEbus, IEEE, New York, 1987

[3] Hassapis G., Simulation of Embedded VME Multiprocessors, Microprocessing and Microprogramming, Vol. 33, pp. 253-260, 1991

[4] Heath S., VMEbus: A Practical Companion, Butterworth-Heinemann, Oxford, 1993

[5] Jain R., The Art of Computer Systems Performance Analysis, John Wiley and Sons, New York, 1991

[6] Kota S.L., J.D. Kallaus, Media Access Protocols for High Speed Data Bus: A Simulation Study, MILCOM'92 Conf. Record, Vol. 3, pp. 1003-1010, IEEE, New York, 1992

[7] McCarron C.W., C.-H. Tung, Simulation Analysis of a Multiple Bus Shared Memory Multiprocessor, Simulation, Vol. 61, No. 3, pp. 169-177, 1993

[8] MODSIM II Compiler. Version 1.9.7, CACI, San Diego, Calif., 1993, `http://www.caciasl.com`

[9] Narasimhan V.L., S. Price-White, Analysis and Simulation of Six Bus Arbitration Protocols, Microprocessing and Microprogramming, Vol. 38, Nos. 1-5, pp. 655-662, September 1993

[10] Obaidat M.S., M.A. Radaieh, A Comparative Simulation Study of the Performance of Single-Bus and Two-Bus Multiprocessors, Simulation, Vol. 56, No. 1, pp. 9-18, 1991

[11] Önyüksel I., Throughput Analysis of Multiple-Bus Multiprocessor Systems with Simultaneous Possession of Common Resources, Simulation, Vol. 61, No. 3, pp. 176-184, 1993

[12] Peterson W.D., The VMEbus Handbook. Second Edition, VITA, Scottsdale, Ariz., 1992

[13] Witten M., B.L. Bodnar, A.C. Liu, Simulation and Modeling of a Single Bus Tightly Coupled Multiprocessor System, Mathematics and Computers in Simulation, Vol. 29, pp. 19-31, 1987

[14] Zalewski J. (Ed.), Advanced Multimicroprocessor Bus Architectrures, IEEE Computer Society Press, Los Alamitos, Calif., 1995

Appendix I

```
CPUObj = OBJECT
       ID : INTEGER;
      maxWaitTime : REAL;
      random : RandomObj;
      MessageLength : INTEGER;
      BytesTransmitted : INTEGER;
      MessageLeft : BOOLEAN;
      InControl : BOOLEAN;
      interarr : REAL;
      access : INTEGER;
       ASK METHOD SetID(IN n : INTEGER);
       ASK METHOD SetAttributes(IN rnd : RandomObj; IN val : REAL);
        ASK METHOD GenerateRequest(IN bus : BusObj);
      WAITFOR  METHOD CommunicateP(IN bus : BusObj);
      WAITFOR  METHOD CommunicateR(IN bus : BusObj);
      TELL METHOD Run(IN ST : REAL; IN bus : BusObj; IN stat: INTEGER);
  END OBJECT;

BusObj = OBJECT(ResourceObj)
  Latency : StatQueueObj;
  Utilization : StatQueueObj;
  Throughput : REAL;
  maxLatency : REAL;
  status : BOOLEAN;
  arbitrationTime : REAL;
  terminationTime : REAL;
  noAccess : INTEGER;
    ByteTransmissionTime : REAL;
        ASK METHOD SetStatus(IN val : BOOLEAN);
        ASK METHOD SetTermination(IN val : REAL; IN mwt : REAL);
        ASK METHOD AddBytes(IN byte : INTEGER);
        ASK METHOD AddRequest();
        ASK METHOD CheckQueue(IN cpin : CPUObj): INTEGER;
OVERRIDE
   ASK METHOD Create(IN number : INTEGER);
END OBJECT;

ReportObj = OBJECT
   Throughput : REAL;
   Latency : REAL;
   MaxLatency : REAL;
   Utilization : REAL;
   noAgents : INTEGER;
   commRate : INTEGER;
   runTime : REAL;
   iteration : INTEGER;
   ASK METHOD SetParameters(IN no : INTEGER; IN rate : INTEGER; IN totTime : REAL);
   ASK METHOD Report(IN bus : BusObj; IN stat : BOOLEAN);
   ASK METHOD ReportGlobal();
END OBJECT;
```

Appendix II

```
 VME BUS SIMULATION: Priority/RoundRobin Case
 Workload: 50%: 4 Bytes, 49%: 100-1000Bytes; 1%: 10,000-100,000Bytes
 Constants:   ByteTransmissionTime = 0.0001 msec;   ArbitrationTime = 0.0005 msec
 ENTER NO OF CPU's
10
 ENTER NO OF ITERATIONS
100
ENTER SIMULATION TIME IN SECONDS (0.5-10)
1
DO YOU WANT DETAILED LONG TRACE ?? (no-1, yes-0)
1
 SINGLE CPU BUS REQUEST RATE PER SECOND (1 - 200, i.e. 1,000  msec - 5 msec)
100
 PRIORITY CASE
======= GLOBAL REPORT OF 100 RUNS ==========

 BUS STATISTICS FOR  10 AGENTS;
 COMMUNICATION RATE 100 per sec   (10.000000 msec)
        AVERAGE BUS UTILIZATION  8.176573 %
        AVERAGE DATA TRANSMITTED  793.037539 KB in 1000.000000 msec
        AVERAGE THROUGHPUT EFFECTIVE RATE  0.793038 MB/sec
        AVERAGE BUS REQUEST LATENCY/CPU  7.259593 msec
        MAXIMAL BUS REQUEST LATENCY  16.330634 msec
================================================

ROUND ROBIN CASE
======= GLOBAL REPORT OF 100 RUNS ==========

 BUS STATISTICS FOR  10 AGENTS;
 COMMUNICATION RATE 100 per sec   (10.000000 msec)
        AVERAGE BUS UTILIZATION  8.208722 %
        AVERAGE DATA TRANSMITTED  795.965449 KB in 1000.000000 msec
        AVERAGE THROUGHPUT EFFECTIVE RATE  0.795965 MB/sec
        AVERAGE BUS REQUEST LATENCY/CPU  5.382323 msec
        MAXIMAL BUS REQUEST LATENCY  16.330634 msec

==============================

BUS STATISTICS FOR  10 AGENTS
        TOTAL SIMULATION TIME: 499.202960 msec
        MAX #AGENTS WAITING FOR ACCESS 1

        MEAN #AGENTS WAITING FOR ACCESS 0.500000

        BUS UTILIZATION  4.349093 %
        TOTAL BUS TRANSMISSION TIME  21.710800 msec

        BUS REQUEST LATENCY/CPU  0.583179 msec
        MAX BUS REQUEST LATENCY  5.668664 msec

        NUMBER OF TRANSMISSIONS  242  in 499.202960 msec
        AGENTS TRANSMISSIONS RATE   48.477277 per sec
        TOTAL DATA TRANSMITTED  210.828125  KB in 499.202960 msec
        THROUGHPUT EFFECTIVE RATE  0.422329 MB/sec
        THROUGHPUT PEAK RATE  9.710749 MB/sec

==================================
```