An Evaluation of the VME Architecture for Use in Embedded Systems Education

Kenneth G. Ricks, David J. Jackson, and William A. Stapleton

Abstract— **The VMEbus is an IEEE standard architecture upon which many embedded and real-time systems are built. It has existed for nearly 25 years and has been extensively used for military, industrial, and aerospace applications. This paper describes the general characteristics of the VMEbus architecture, specifically relating these characteristics to aspects of embedded systems education included as components of the IEEE/ACM CE2004 computer engineering model curriculum. Portions of this model curriculum are currently being implemented at universities across the country as part of an increasing effort to address the need for embedded systems education. This evaluation will help to identify the strengths and weaknesses of this architecture as a general-purpose embedded systems educational tool.**

Index Terms— **Computer architecture, computer engineering education, educational technology, embedded systems.**

I. Introduction

EMBEDDED systems are becoming more and more common as feature size and power requirements of electronic components decrease and overall performance improves. As a result, more and more graduates are entering careers working directly with the design and implementation of embedded systems [1]. In order to prepare graduates for the challenges in this field, many universities are incorporating embedded systems concepts into their computer engineering curriculum. However, embedded systems education is difficult to generalize due to the following:

- Embedded systems education incorporates a broad range of concepts from many disciplines;
- Different programs have different goals associated with their embedded systems curricula;
- There exists a diverse range of embedded applications;
- There are many different technologies used in embedded systems;
- There is a wide range of design constraints imposed on embedded systems; and
- There are many different embedded systems architectures and platforms in use today.

Because of these characteristics, embedded systems education is fragmented and customized from one university to another. In an attempt to provide curricular guidelines for education in this field, the IEEE/ACM CE2004 computer engineering model curriculum for embedded systems was created [2]. This model provides a comprehensive list of recommended topics that are important for embedded systems education. Universities are challenged to integrate these topics into the classroom and to create laboratory platforms capable of supporting these topics.

In this paper, we evaluate a popular embedded systems architecture to determine its suitability as a general-purpose laboratory platform for embedded systems education. This architecture, the VMEbus, was developed nearly 25 years ago and has been used extensively in industrial, military, aerospace, communication, and control applications. Despite its age, it is still one of the most popular architectures for embedded and real-time systems accounting for 38% of the embedded systems market in 2002 [3,4]. The emphasis here will be to evaluate the VMEbus architecture against the model curriculum as opposed to comparing various embedded architectures.

The remainder of this paper is organized as follows. Section II provides a general description of the VMEbus architecture. Section III evaluates how the architecture supports many of the components of the model curriculum. Conclusions are presented in section IV.

II. The VMEbus Architecture

A. Background

The VMEbus is a system architecture consisting of the electrical specifications for a communication bus and the mechanical specifications describing the backplane, bus connectors, board sizes and formats, as well as card cages, racks, and enclosures. VME stands for Versa Module

This work was supported in part by the National Science Foundation under grants DUE-0310831 and EEC-0431792.

Kenneth G. Ricks is with the Department of Electrical and Computer Engineering, The University of Alabama, Tuscaloosa, Alabama, 35487-0286 USA. (phone: 205-348-9777; fax: 205-348-6959; e-mail: kricks@coe.eng.ua.edu).

David J. Jackson is with the Department of Electrical and Computer Engineering, The University of Alabama, Tuscaloosa, Alabama, 35487-0286 USA. (phone: 205-348-2919; fax: 205-348-6959; e-mail: jjackson@eng.ua.edu).

William A. Stapleton is with the Department of Electrical and Computer Engineering, The University of Alabama, Tuscaloosa, Alabama, 35487-0286 USA. (phone: 205-348-1436; fax: 205-348-6959; e-mail: wstapleton@eng.ua.edu).

Eurocard and refers to the VERSAbus, the predecessor electrical data bus standard, and the Eurocard, a predecessor mechanical format for computer components. The VME specification was completed in 1982 and is now part of the ANSI/IEEE STD. 1014-1987. This is an open architecture allowing third parties to freely develop VMEbus products [5, 6].

B. General VMEbus Architecture

The VMEbus architecture is a shared system bus architecture. The system bus resides on a backplane. The backplane has slots where processor modules, memory modules, or I/O modules connect to the system bus [6, 7]. The backplane and all of the modules connected to it reside in an enclosure that serves to protect the components, provide structural support for the system, and house utility components such as power supplies and cooling fans. Figure 1 shows one possible system configuration. The modular design of this architecture provides great flexibility. All the components are available commercially-off-the-shelf (COTS).

C. System Bus Characteristics

A functional block diagram of the VME system bus is given in Figure 2, and the basic system bus characteristics are summarized in Table 1. It is beyond the scope of this paper to describe all of the specific characteristics and the timing behavior of the bus. However, Figure 2 and Table 1 provide a broad summary of the basic bus characteristics that will aid the reader during this evaluation. Also, these characteristics are those of the basic VME32 specification. Many extensions to this specification have been created, but for this discussion, the basic design will be used.

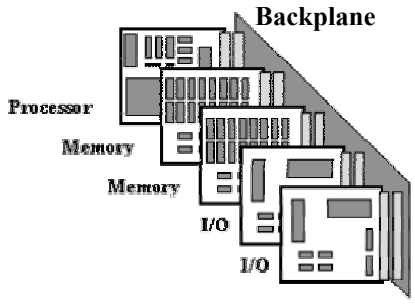

Figure 1. Diagram of the general components and configuration of the VMEbus architecture [8].

Table 1. Summary of the VME32 system bus characteristics [5, 6].

Architecture Master/Slave
Transfer Mechanism Asynchronous (no central synchronization clock) Fully handshaked Non-multiplexed Multiplexed for 64-bit transfers
Addressing Range 16-bit (short I/O) 24-bit (standard I/O) 32-bit (extended I/O) 64-bit (long I/O) Address range is selected dynamically
Datapath Width 8, 16, 24, 32, 64-bit Datapath width selected dynamically
Unaligned Data Transfers Yes Compatible with most popular processors
Error Detection Yes Using BERR* (optional)
Data Transfer Rate 0-80 Mbytes/sec
Interrupts 7 levels Priority interrupt system with STATUS/ID
Multiprocessing Capability 1-21 processors Flexible bus arbitration
System Diagnostic Capability Yes Using SYSFAIL* (optional)
Mechanical Standard Single height (160 x 100 mm eurocard) Double height (160 x 233 mm eurocard) DIN 603-2 connectors

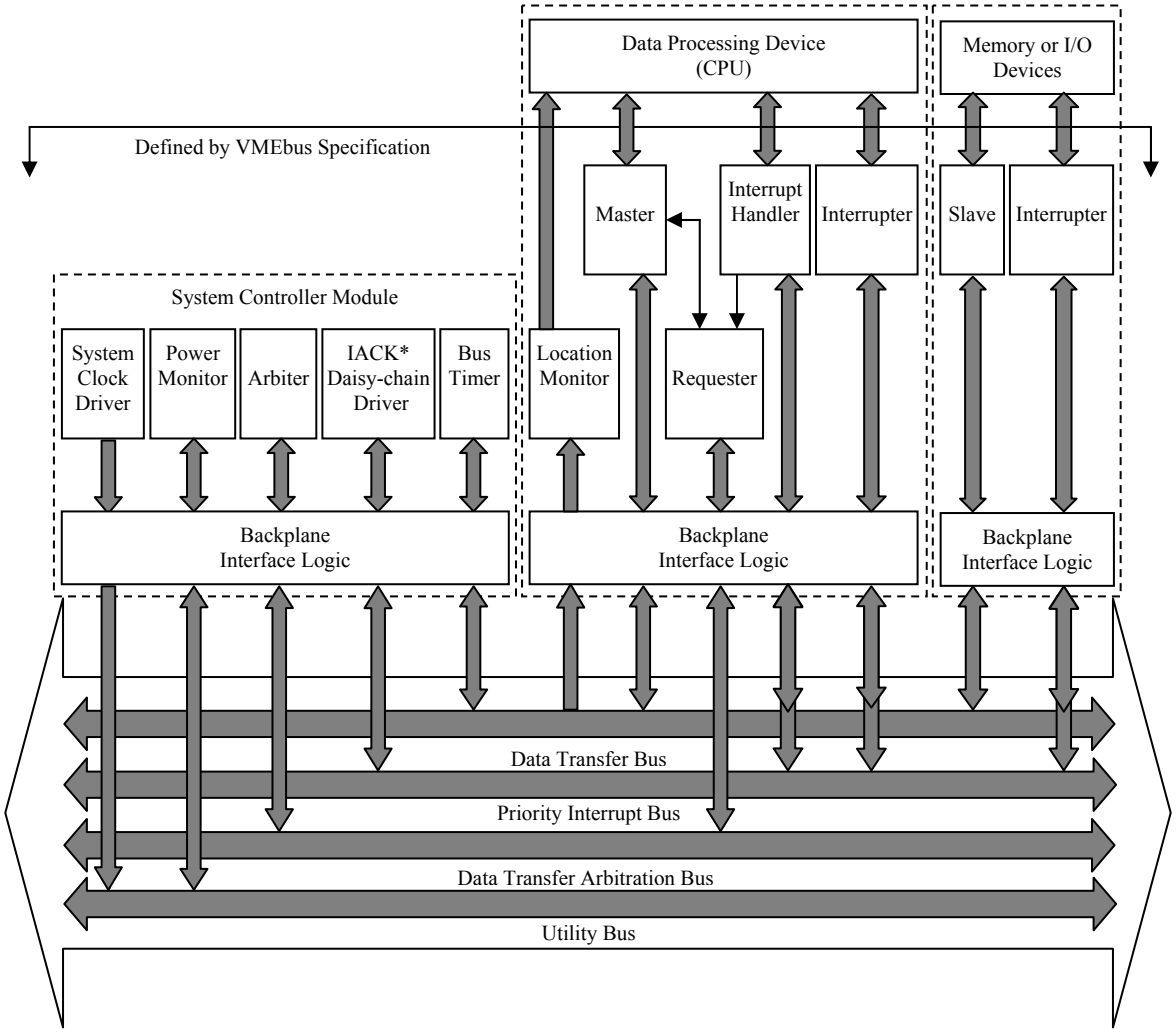

Figure 2. Functional block diagram of the VME system bus [6].

III. The VMEbus Support for the Model Curriculum

To evaluate the capability of the VMEbus architecture for use as a general-purpose embedded systems educational platform, its characteristics and capabilities will be discussed as they apply to each embedded systems component of the IEEE/ACM model curriculum for computer engineering [2]. The rationale is that if the architecture can address the concepts in the model curriculum it should be able to support the diverse embedded systems curricula being taught across the country.

The embedded systems portion of the model curriculum is broken into eleven areas of which seven are core topics and four are considered to be elective material. In the following subsections, each of these areas is addressed. Each section begins with a list of the topics and learning outcomes from the model for a particular area that can be supported by the VMEbus architecture. How the architecture supports these topics and learning outcomes for each area is then presented.

A. CE-ESY0 History and Overview (core)

Topics

- Highlight some people that influenced or contributed to the area of embedded systems;
- Contrast between an embedded system and other computer systems [2].

Learning Outcomes

- Identify some contributors to embedded systems and relate their achievements to the knowledge area;
- Describe the meaning of an embedded system;
- Explain the reasons for the importance of embedded systems [2].

The VMEbus architecture holds its own place in the history and evolution of embedded systems. It was developed specifically for embedded applications. It is one of the most frequently used architectures for real-time applications [3] and accounts for 38% of the embedded systems market [4]. Also, the fundamental components of the VMEbus architecture are designed and configured differently than those used in desktop computer systems. The form factor, component

placement, component interaction, and overall system design is different. This makes it easy to introduce embedded systems, compare and contrast them to general-purpose computing systems, and to justify their significance.

B. CE-ESY1 Embedded Microcontrollers (core)

Topics

- Structure of the basic computer system: CPU, memory, I/O devices on a bus;
- CPU families used in microcontrollers: 4-bit, 8-bit, 16-32-bit;
- Basic I/O devices: timers/counters, GPIO, A/D, D/A;
- Polled I/O vs. interrupt-driven I/O;
- Interrupt structures: vectored and prioritized interrupts;
- Direct memory Access (DMA) transfers;
- Memory management units;
- Memory hierarchies and caches [2].

Learning Outcomes

- Understand the CPU in the context of a complete system with I/O and memory;
- Understand how the CPU talks to the outside world through devices;
- Understand how memory system design (caches, memory management) affect program design and performance [2].

The VMEbus architecture design is based upon the most basic of system structures. The modular design and interchangeable characteristics of the system components gives students real system design and integration experience reinforcing the first learning outcome. Also, the asynchronous bus protocol and variable width data transfers make the architecture compatible with many different processor families. This is a key concept since a large percentage of embedded systems rely on technology other than 32-bit processors [9]. The ability to connect a logic analyzer directly to the bus signals also provides a means for students to analyze bus and system timing, to study the asynchronous handshaking protocol in depth, and to investigate bus arbitration and bus saturation.

Since the architecture was originally designed for I/O intensive applications, there are many COTS I/O devices that can be used to teach the I/O topics and achieve the second learning outcome. The asynchronous system bus design allows the architecture to support I/O peripherals of various speeds using either polled I/O or an interrupt-driven design. The VME system bus supports 7 levels of prioritized interrupts and can handle vectored interrupts.

The VME specification also provides flexible memory design helping to achieve the third learning outcome in this area. For example, single transfers can be used or block transfers can be used for DMA operations. Different memory configurations can be created ranging from shared global memory to completely distributed memory creating both uniform memory access (UMA) and non-uniform memory access (NUMA) system configurations [10]. Cache memory is commonly available on processor modules and contributes to the design of various memory hierarchies. This also allows for the introduction of memory inconsistency into the curriculum and provides a platform to demonstrate its effects on system and program performance.

C. CE-ESY2 Embedded Programs (core)

Topics

- The program translation process: compilation, assembly, linking;
- Fundamental concepts of assembly language and linking: labels, address management;
- Compilation tasks: mapping variable to memory, managing data structures, translating control structures, and translating expressions;
- What can/cannot be controlled through the compiler; when writing assembly language makes sense [2].

Learning outcomes

- Understand how high-level language programs convert into executable code;
- Know the capabilities and limits of compilers [2].

The VMEbus architecture supports these topics and learning outcomes through the wide variety of operating systems and software development environments available for use with the architecture. The variety of software available for this architecture is unparalleled. For example, there are at least 103 operating systems known to be ported to this architecture [11]. Compilers and assemblers for many different high-level languages (HLLs), such as C, C++, and FORTRAN, and assembly languages are available depending upon the system components used. This makes it convenient to examine high-level code and its associated lower-level constructs. Applications can be implemented in both a HLL and an assembly language for performance comparisons. Also, the processor modules can be used without an operating system (OS), which is often the case for embedded systems. In this case, students are exposed to rudimentary development environments possibly even entering the binary representation of each instruction directly into memory manually. Students can be exposed to both locally hosted development environments and cross-compiled environments.

D. CE-ESY3 Real-Time Operating Systems (core)

Topics

- Context switching;
- Real-time scheduling concepts;
- Interprocess communication mechanisms [2].

Learning outcomes

- Distinguish a real-time operating system (RTOS) from a workstation/server OS;
- Distinguish real-time scheduling from traditional OS scheduling;
- Understand major real-time scheduling policies;
- Understand interprocess communication mechanisms [2].

The VMEbus architecture is one of the most popular

architectures for real-time computing and has been for many years [3]. Many different RTOSs have been ported to many different processor families for this architecture. Some of these RTOSs include Real-Time Linux, VxWorks, QNX, and OS-9. Many of these RTOSs support the POSIX real-time extensions for interprocess communication. Various scheduling strategies and timing issues can be investigated. A VMEbus system using an RTOS can be configured to address an exact subset of these topics and learning outcomes to accommodate different educational strategies. In addition, multiple processor modules in one VMEbus system can be configured to execute different OSs. In this way, direct comparisons can be made between an RTOS and a non-real-time OS executing on the exact same architecture.

E. CE-ESY4 Low Power Computing (core)

Topics

- Sources of energy consumption;
- Instruction-level strategies for power management: function unit management;
- Memory system power consumption: caches, off-chip memory;
- Power consumption with multiple processes;
- System-level power management: deterministic, probabilistic methods [2].

Learning outcomes

- Understand why low-power computing is important;
- Identify sources of energy consumption;
- Identify possibly remedies for energy consumption at various levels of design abstraction [2].

There is no specific support in the VMEbus specification for low power computing. A typical VMEbus enclosure is not remotely powered, thus eliminating the need for strict energy management strategies such as those seen in handheld and remotely deployed embedded systems. However, this does not prevent one from monitoring energy consumption of the system modules such as memory and comparing and contrasting energy usage in single and multiple processor systems to achieve the learning outcomes. Additionally, the asynchronous bus protocol will support both high and low speed processors in the same system. Since reducing clock speed is one primary power management technique, this architecture can provide a platform to make direct comparisons of power consumption for system components having different clock speeds. Finally, custom low power circuitry can be developed on prototyping system modules and interfaced to the VMEbus system for control and monitoring. Controlling custom low power components within an existing system architecture can also help to achieve the learning outcomes suggested in this area of the model curriculum.

F. CE-ESY5 Reliable System Design (core)

Topics

- Transient vs. permanent failures in hardware;
- Sources of errors from software;
- The role of design verification in reliable system design;
- Fault-tolerant techniques [2];

Learning outcomes

- Understand the variety of sources of faults in embedded computing systems;
- Identify strategies to find problems;
- Identify strategies to minimize the effects of problems [2].

To address these concepts and learning outcomes, students using the VMEbus architecture can be exposed to a wide variety of problems ranging from low-level errors in the fundamental bus transactions to debugging software written in high-level languages. Fault tolerance using redundancy can be introduced by including redundant system components. Students have access to both hardware and software on these systems requiring failure strategies and debugging in both domains.

G. CE-ESY6 Design Methodologies (core)

Topics

- Multi-person design projects;
- Designing on-time and on-budget;
- Design reviews;
- Tracking error rates and sources;
- Change management [2].

Learning outcomes

- Understand why real-world projects are not the same as class projects;
- Identify important goals of the methodology;
- Understand the importance of design tracking and documentation [2].

The VMEbus architecture actually supports these design concepts and learning outcomes better than many other educational technologies. The reason for this is that the VMEbus architecture supports design at various levels of abstraction. For example, given a set of modules, students can make meaningful system-level design decisions including the number of processors, the design of the I/O system, and the configuration of memory (distributed, shared, hierarchy characteristics, etc.). Low-level design can be incorporated into a project by having students design logic components capable of interfacing to the VME system bus. Real-world concepts can be addressed by incorporating VMEbus component documentation into projects, thus exposing students to typical technical documentation they are likely to see as a practicing engineer. This tends to reinforce the importance for students to produce accurate documentation of their own systems. The fact that the VMEbus is a real embedded systems architecture can motivate students since they see the project as more than an academic exercise.

H. CE-ESY7 Tool Support (elective)

Topics

- Compilers and programming environments;

- Logic analyzers;
- RTOS tools;
- Power analysis;
- Software management tools;
- Project management tools [2].

Learning outcomes

- Understand the role of hardware and software tools in system design;
- Understand how to use tools to support the methodology [2].

The tools and environments available to VMEbus designers are plentiful and varied in design, objective, and scope. There are many different software tools such as compilers, assemblers, debuggers, development environments, and real-time analysis tools. There are also custom function libraries and drivers tailored to particular system components available from various vendors. For locally hosted systems, software tool availability is dependent upon the OS executing on the processor module. But, this is typically not a limitation since, as previously mentioned, at least 103 OSs have been ported to this architecture [11]. There are also special purpose logic analyzers designed specifically to analyze VME system bus timing. Individual institutions can select particular OSs and tools to tailor their VMEbus system to accommodate their particular curricular goals and objectives. Finally, something that is not directly listed in the model curriculum but does represent a major concept is the development of device drivers [12]. If desired, students can develop drivers for various I/O peripherals using this architecture.

I. CE-ESY8 Embedded Multiprocessors (elective)

Topics

- Importance of multiprocessors as in performance, power, and cost;
- Hardware/software partitioning for single-bus systems;
- More general architectures;
- Platform field programmable gate arrays (FPGAs) as multiprocessors [2].

Learning outcomes

- Understand the use of multiple processors in embedded systems;
- Identify trade-offs between CPUs and hardwired logic in multiprocessors;
- Understand basic design techniques [2].

With the increasing complexity of embedded systems, multiprocessing is becoming a requirement in order to meet performance specifications. This is especially true for real-time applications. One of the strengths of the VME architecture is its multiprocessing capabilities [4]. The modular design of the components makes it easy to create various multiprocessor configurations. For example, all processors can be identical creating a homogeneous system, or each processor can be different, even executing different OSs, to produce a heterogeneous system. Memory configurations are also flexible ranging from completely shared to completely distributed. Thus, the VMEbus architecture can easily be used to teach students general multiprocessor architectures and basic design techniques. FPGA devices integrated into a VMEbus module are available and can be used to teach hardware/software co-design [13], partitioning, trade-off analysis and performance analysis. The use of reconfigurable hardware technology for embedded systems education is common [14], but incorporating it into the VMEbus architecture creates a robust educational platform where this hardware can be integrated with real-time system components. Multiprocessor architectures can be used to teach other topical areas in the model curriculum as well including bus characteristics such as arbitration and saturation. Bus saturation is a key design consideration within multiprocessor architectures. Also, memory characteristics such as mutual exclusion and interprocess communication, as discussed in section III *D,* are reinforced.

J. CE-ESY9 Networked Embedded Systems (elective)

Topics

- Why networked embedded systems;
- Example networked embedded systems;
- The OSI reference model;
- Types of network fabrics;
- Network performance analysis;
- Basic principles of the Internet protocol;
- Internet-enabled embedded systems [2].

Learning outcomes

- Understand why networks are components of embedded systems;
- Identify roles of hardware and software in networked embedded systems;
- Compare networks designed for embedded computing with Internet networking [2].

The VMEbus architecture can provide insight into networking concepts in several different ways. Since the architecture is common in industrial control applications, it is normal to have VMEbus enclosures physically distributed throughout the factory yet logically connected to promote sharing of resources and provide for centralized monitoring and control. There are many different products available that can logically connect distributed VMEbus systems. Some of these products include reflective memory (shared memory), fiber optics, and various standardized communication protocols like the TCP/IP and the MIL-STD-1553 bus standard. With so many choices, institutions can investigate concepts such as shared memory, dedicated local-area-networks, shared wide-area-networks, different protocols, and network bandwidth and congestion. Comparisons can be made between dedicated networks and Internet networking. Various customized networked systems can be created to achieve the specified learning outcomes.

K. CE-ESY10 Interfacing and Mixed-Signal Systems (elective)

Topics

- Digital-to-analog (D/A) conversion;
- Analog-to-digital (A/D) conversion;
- How to partition analog/digital processing in interfaces;
- Digital processing and real-time considerations [2].

Learning outcomes

- Understand pros and cons of digital and analog processing in interfaces;
- Understand fundamentals of A/D and D/A conversion [2].

A/D and D/A conversion are important components of I/O processing and are addressed in that context in section III *B*. The main topic in this section deals with mixed-signal design. There is no direct support for this within the VMEbus specification. However, mixed-signal design fits well within the VMEbus architecture which was designed for I/O intensive applications. Specifically, interfacing the digital and analog components of the system can be addressed using bus modules which include prototyping areas for custom design. In this way, students can be tasked with interfacing analog components to an existing digital system outside the normal I/O context. The real-time aspects of the architecture provide unlimited possibilities for investigating signal processing with real-time constraints.

IV. CONCLUSIONS

Embedded systems education is rapidly being implemented in electrical and computer engineering programs across the nation. But, it is unlikely that institutions of higher learning will implement a common embedded systems curriculum in the near future. The broad nature of the field and different educational objectives have led to highly customized curricula supported by equally customized laboratory platforms. The IEEE/ACM model curriculum attempts to provide guidelines for the concepts required in this field. Laboratory platforms capable of supporting these concepts can be applied as general-purpose educational tools across many curricula.

In this paper, the VMEbus architecture is evaluated against the model curriculum to determine its suitability as a general-purpose educational platform in this field. This architecture is shown to support many of the concepts in the model curriculum addressing many different educational objectives and learning outcomes. Specifically, there is strong support for topics including historical and introductory concepts, system design, I/O, real-time systems, multiprocessing, memory configurations, networked embedded systems, and software tool support. Other benefits of the use of this architecture include exposing students to a real-world embedded architecture, the wide range of COTS components available supporting various system configurations, and its longevity in the field.

Like all other architectures, it is not perfect. The VMEbus architecture does not address one key concept from the model curriculum, low-power computing. Also, size constraints are not addressed outside the context of the VME form factor. Although size is not specifically part of the model curriculum, system-on-a-chip implementations are critical for many embedded applications including handheld consumer electronics. Finally, VME system components are relatively expensive. This is a serious consideration for educational use.

Despite these weaknesses, the authors feel that the VME architecture represents a powerful, capable embedded systems architecture that is appropriate for general-purpose embedded systems education based upon its strong support for most of the concepts in the model curriculum.

REFERENCES

[1] Y. H. Lee, A. Oo, "Teaching Microprocessor System Design Using a SoC and Embedded Linux Platform", Proceedings of the 2005 Workshop on Computer Architecture Education (WCAE) held in conjunction with the 32nd International Symposium on Computer Architecture, Madison, Wisconsin, June 5, 2005.

[2] Joint Task Force on Computer Engineering Curricula, IEEE Computer Society, Association for Computing Machinery, "Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering", December 12, 2004, pp. A.43 – A.45, Available: http://www.computer.org/education/cc2001/CCCE-FinalReport-2004Dec12-Final.pdf.

[3] A. Kornecki, H. Wojcicki, L. Peltier, J. Zalewski, N. Kruszynska, "Teaching Device Drivers Technology in a Real-Time Systems Curriculum";
Proceedings of the 1998 Real-Time Systems Education III, Nov. 21, 1998, pp. 42 – 48.

[4] J. Moll, "Shared Bus Versus Switched Fabric Technologies", EE Times, January 17, 2003, Available online: http://www.eetimes.com/story/OEG20030116S0036.

[5] *IEEE Standard for a Versatile Backplane Bus: VMEbus*, ANSI/IEEE Standard 1014-1987.

[6] W. D. Peterson, *The VMEbus Handbook3 3rd Edition*, Scottsdale, Arizona: VFEA International Trade Association, 1993.

[7] K. G. Ricks, "An Improved Bus-Based Multiprocessor Architecture", M.S. thesis, Electrical and Computer Engineering Dept., the University of Alabama in Huntsville, Huntsville, Alabama, 1997.

[8] M. Timmerman, "The Right Bus in the Right Place: A Tutorial (part 1)", Real-time Magazine, 4Q96, pp. 6-11, Available: http://www.realtime-info.be/magazine/index/index964.htm.

[9] P. Koopman, "Embedded Systems Design Issues (the Rest of the Story)", Proceedings of the 1996 IEEE International Conference on Computer Design: VLSI in Computers and Processors, October 1-9, 1996, pp. 310-317.

[10] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw-Hill, New York, New York, 1993.

[11] Portions of this FAQ have been reprinted (with permission) from *The VMEbus Handbook, 4th Edition* by Wade D. Peterson. VITA 1997. For more details the user is directed to the handbook, or the VMEbus specification(s). Other items have been reprinted from the VITA Journal (with permission) *VMEbus FAQ's* article series by John Rynearson. Available online: http://www.vita.com/vmefaq/

[12] A. Kornecki, H. Wojcicki, J. Zalewski, N. Kruszynska, "Teaching device drivers technology in a real-time systems curriculum" Proceedings of the 1998 Real-Time Systems Education III, Nov. 21, 1998, pp. 42 – 48.

[13] W. Wolf, Computers as Components: Principles of Embedded Computing System Design, Morgan Kaufmann, New York, New York, 2001.

[14] Proceedings of the 2005 International Conference on Microelectronic Systems Education, Anaheim, California, June 12-13, 2005.