

Design of an Asynchronous VME bus Controller for Heterogeneous systems

We describe the design of an asynchronous VME bus controller to be used with heterogeneous systems. This controller allows to VME bus based systems to communicate with the most known standard mediums and to resolve access conflict problems whenever there is a shared medium contention. Such a medium can be the parallel PCI bus, the serial SCSI bus, the DRAM, the DMA, the 4-phase protocol, and the FIFO protocol. Starting from the HDL descriptions of the VME bus and the communication medium's protocols, we generate a purely behavioural HDL description of the controller. After being generated, the latter description is transformed into an asynchronous synthesis model called extended-burst-mode machine to be implemented by commercial synthesis tools. This design was simulated correctly with respect to all of the applicable test vectors and implemented by using a 0.6 μ m CMOS standard cell technology.

INTRODUCTION

Today embedded computer systems have become everyday gadgets for most people, who use them without even knowing it. The following bulleted items show some examples of things; token from the author's daily life that contains one or more embedded computer systems. Each example includes a description of tasks that can be carried out by embedded systems.

- Audio equipment - providing the user interface and processing the audio data.
- Cellular phone - providing the user interface, encoding data, speech compression, etc.
- TV set-top box - controlling the on-screen user interface, image processing, etc.

One reason why embedded computer systems are so popular is the combination of high-performance hardware with flexible software that has a short design time.

An embedded system has several components, but the most important device, which has enabled the evolution of embedded systems, is the microprocessor [1]. The complexities of the today processors with 6,5 to 100 million transistors (e.g. AlphaTM 21364, Pentium III/TM, etc.) together with memory devices containing up to 256 million devices on a single chip, have made it possible for chips to contain entire systems. This in turn enables system functionality well beyond traditional ASIC-based systems, for example, a whole GSM mobile phone can now be integrated on a single chip.

The modules that constitute an embedded system often interact and communicate in complex ways by transferring information and synchronizing their inputs and outputs. A key bottleneck in system integration is the design of system interface modules, which are special modules required for interconnecting and synchronizing the information transfer between communicating system-level components. This is required because in general each of the communicating components may use arbitrary and incompatible I/O protocols to communicate with its environment.

Originally introduced in 1981 [2], VME bus has become the standard high performance bus structure. It is the bus of choice for the present and future of microcomputer architecture, supporting data transfer rates as high as 24 Mbytes in the expanded 32-bit configuration. This bus has a master slave asynchronous non-multiplexed data transfer structure, seven levels of priority interrupt, four levels of data bus arbitration and rapid fault detection and control for bus, system and AC failures.

In this paper, we describe the design of an asynchronous VME bus controller (Figure 1) which is used as the interface between a VME bus and an external medium that implements a standard communication protocol. Such a medium can be the parallel PCI bus, the serial SCSI bus, the DRAM, the DMA, the 4-phase protocol, and the FIFO protocol. In fact, the latter mediums may be the most popular protocols that are used for heterogeneous system communication. While the VME bus is designed with bus arbitration scheme

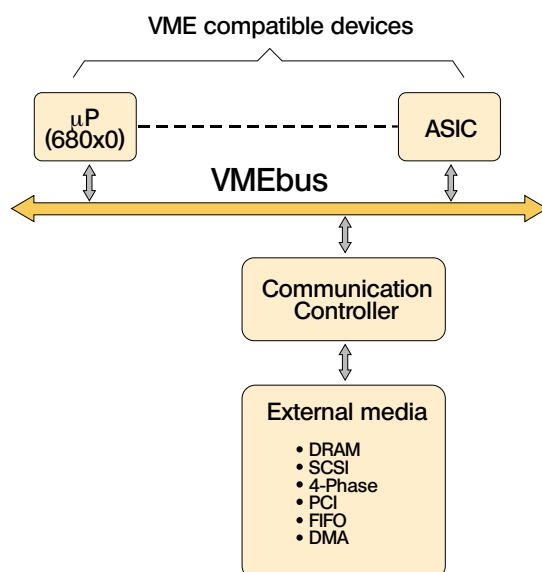


Figure 1. A VME bus based system overview.

AD EONIC

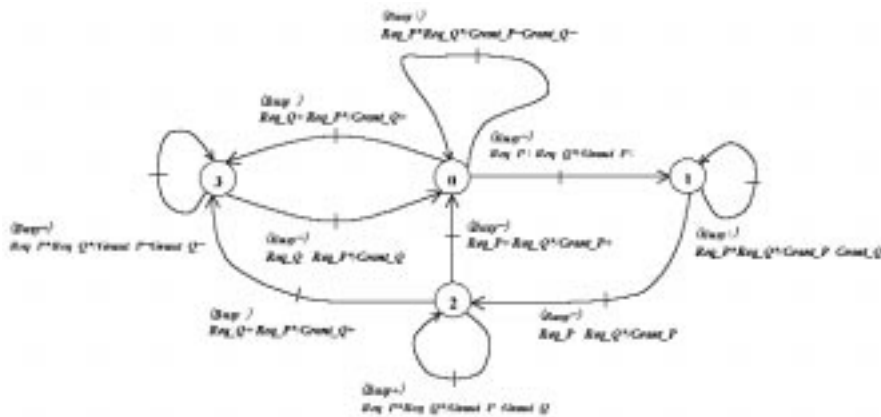


Figure 2. Extended burst-mode-machine of a round-robin arbiter with two requesters.

incorporated in its protocol, the communication controller contains an arbiter module that allows resolving access conflict problems whenever there is a shared medium contention in the system.

The remainder of this paper is organized as follow. In section 2, we describe the design of the controller. Section 3 presents some experiment results related to the controller. In section 4, we conclude the paper.

VME BUS CONTROLLER

The most used standard protocols for communication into a distributed system are generally asynchronous bus (e.g. VME bus), Parallel bus (e.g. PCI bus), Serial bus (e.g. SCSI bus), 4-phase handshake protocol, the burst transfer (e.g. DMA), the blocking communication (e.g. FIFOs) and the shared memory based communication. The controller is constituted by a set of I/O blocks that allow to a VME bus based system to communicate with an external medium that integrates a given protocol among the indicated protocols. In the case of a multiprocessor VME bus based system, the external medium may be a shared resource for the system masters. For this reason we have integrated into the controller, rather than the communication module, another module for arbitrating shared medium access. Due to the controller, the addition of other peripheral sub-systems or microprocessor based sub-systems that integrates the indicated protocols without modifying the actual VME bus based architecture may be possible.

ARBITER MODULE

The controller arbiter module allows to dynamically allocating a shared external medium to the master modules in the VME bus based system without a common clock. This allocation is performed by executing a given arbitration scheme based on the VME bus arbitration protocol. Thus it allows the resolution of access conflict problems whenever there is a shared medium contention in the system. This module integrates the fixed and the round-robin priority arbitration schemes. The arbitration protocol executed by this module is described as follow.

- Each requester in the system, when it needs the shared external medium issues an asynchronous

request and waits until the arbiter produces a grant.

- The arbiter, when it receives a number of active requests from different requesters and if the bus is not busy (address strobe, data transfer acknowledge, and bus grant acknowledge are negated) generates after some delay required by the arbitration scheme a grant to exactly one of them and leaves other requests pending until the granted requester released the request.
- The requester then uses the external medium and after finishing its action releases its request. These results in a subsequent release of the grant, after which the requester can issue another request and so on.
- The arbiter releases the grant and, if there are pending requests, produces another active grant, again on fixed or round robin priority basis.

We have described the arbiter module by using the VHDL language [3] at RTL level starting from the extended-burst-mode machine specification model that we have generated automatically starting from purely behavioural non-synthesizable HDL description [4]. The extended-burst-mode machine of a round robin arbiter with two requesters (P and Q) is presented in figure 2.

In an extended-burst-mode machine [5], signals not enclosed in angle brackets and ending with + or - are terminating signals. These are edge signals. The signals enclosed in angle brackets are conditionals, which are level signals whose values are sampled when all of the terminating edges associated with them have occurred. A conditional $\acute{a}a+\bar{n}$ can be read if a is high and $\acute{a}a-\bar{n}$ can be read if a is low. A state transition occurs only if all of the conditions are met and all the terminating edges have appeared. A signal ending with an asterisk is a direct don't care. If a is a direct don't care, there must be a sequence of state transitions in the machine labelled with a*. If a state transition is labelled with a*, the following state transitions in the machine must be labelled with a* or with a+ or a- (the terminating edge for a direct don't care). Figure 2 describes a machine having one conditional input (Busy), 2 edge inputs (Req_P and Req_Q), and 2 outputs (Grant_P and Grant_Q). Consider the state transi-

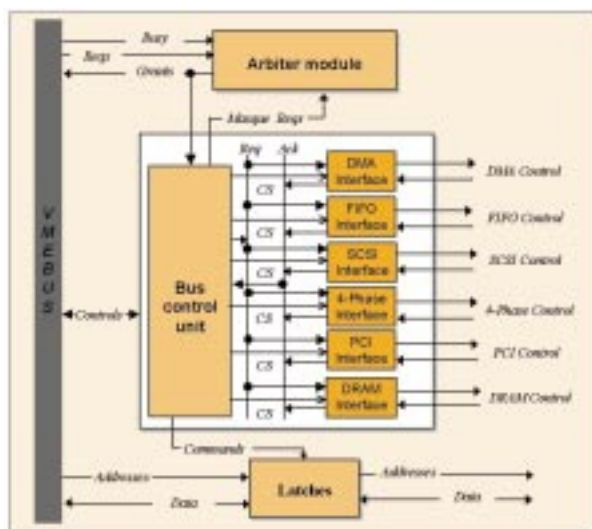


Figure 3. Communication module architecture.

tion out of state, State_0. The behavior of the machine at this point is: if Busy is low changes the current state from State_0 to State_1 and higher the output Grant_P when Req_P rises; changes the current state from State_0 to State_3 and higher the output Grant_Q when Req_Q rises. If Busy is high remains in State_0 and lower the outputs Grant_P and Grant_Q.

COMMUNICATION MODULE

The architecture of the communication module (Figure 3) is constituted by several adaptation interfaces and a bus control unit. The adaptation interfaces allow the I/O communication with the external mediums.

These interfaces communicate with the bus control unit by using a classical 4-phases handshake protocol. After being selected, a given adaptation interface waits an event on the Req line in order to control the communication. The data are transferred through external latches. The latter's are activated in the desired time by the control unit. After terminating the commu-

nication control, the selected interface activates the Ack line. The others interfaces are not selected and they must put the Ack line in high impedance while protecting it from an eventual conflict. After receiving the response on the Ack line, the control unit deactivates the Req line and wait until the deactivation of the former. Thus it deactivates the selected interface. Finally, bus control unit waits until the deactivation of the Grants lines. In fact, the bus control unit is activated only when it receives a rising edge on a given Grant line generated by the arbitrator module indicating that there is a master that gains the access to the shared medium.

The control unit depending on the address transmitted on the system bus by the master that gains the access to the shared medium performs the decision of selecting a communication interface. During the activation of the DMA interface, data transfer between the shared memory and the DMA controller are performed by the system bus. In this case the DMA interface becomes the bus master. For this reason, the bus control unit sends a Masque-Reqs signal to the arbitrator module. When receiving the latter signal, the arbitrator module doesn't respond to requesters. The Mask-Reqs signal is deactivated since the current transfer is terminated and the system bus is on high impedance.

HIGH LEVEL DESIGN OF THE COMMUNICATION MODULE

As indicated above, the communication module is constituted by a set of communication blocks. Each block constitutes an adaptation interface that converts the VME bus protocol to the protocol of each external medium.

Starting from the HDL descriptions of the two incompatible protocols detailing the number of control and data lines and the sequence of data transfers over those lines, we generate a HDL description of the interface that adapt the two protocols. The HDL descriptions of the protocols are captured from the timing diagrams as depicted in figure 4.a and figure 4.b in the

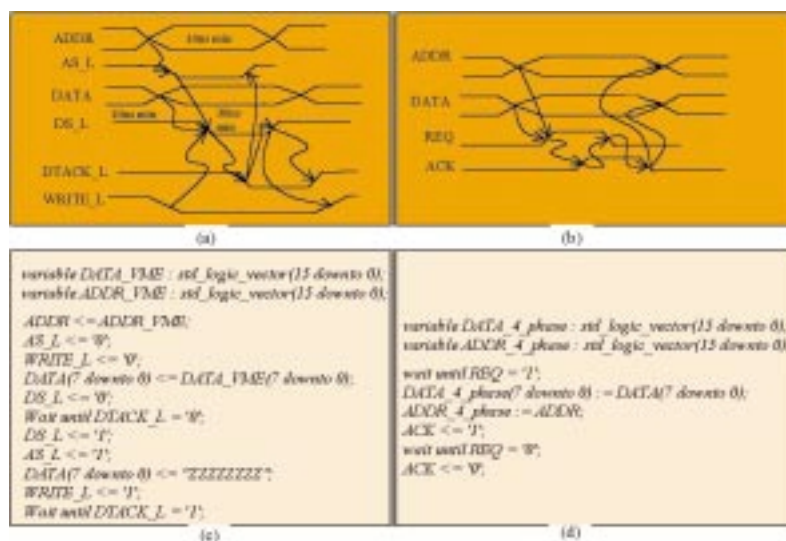


Figure 4. Interface between the VME bus low byte write protocol and the 4-phase protocol.

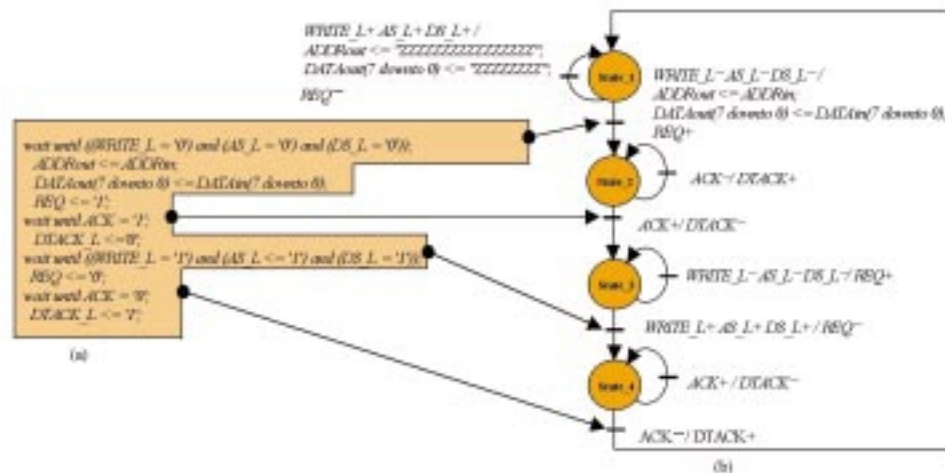


Figure 5. HDL description and extended burst-mode-machine of the VME bus low byte write protocol and the 4-phase protocol adaptation interface.

case of the VME bus low byte write (Figure 4.c) and the 4-phase (Figure 4.d) protocols. The protocols shown are somewhat simplified to clarify the example. The generation of the adaptation interface (Figure 5.a) is performed automatically by using the interface process generation technique [5]. After being generated the HDL description of the adaptation interface is transformed into an extended-burst-mode machine (Figure 5.b) by using our RTL interface synthesis technique [4]. The generated machine is transformed into a VHDL based description to be synthesized by a commercial synthesis tool.

RESULTS

The design is simulated and synthesized using Synopsys. The resulting netlist is used as input to Cadence in order to perform mapping and routing with a 0.6µm CMOS technology. The design results are presented in table 1 and figure 6.

The VHDL description of the design is about 3000 lines of code at RTL level. The final ASIC has been implemented using a CMOS 0.6µm technology and its area is about 10mm². This circuit integrates 6000 equivalent gates.

CONCLUSION

Designers of task-specific systems must deal with a wide collection of interfaces. Applications range from medical instrumentation to communication and networking devices to controllers in automobiles.

We designed an asynchronous VME bus controller, which can be used in order to allow the VME bus based multiprocessor systems to communicate with the most popular external mediums. Such a medium can be the parallel PCI bus, the serial SCSI bus, the DRAM, the DMA, the 4-phase protocol, and the FIFO protocol.

The design of the controller has been performed starting from a purely behavioral non-synthesizable HDL description. By transforming the input description we have generated an RTL description to be synthesized by commercial synthesis tools.

The completed design was mapped to a 0.6µm CMOS standard cell library, simulated and synthesized with Synopsys tool ■

Abdelkrim ZITOUNI is a Ph. D. student at Faculty of Sciences, Monastir, Tunisia. He is a member of

Controller units	Number of gates	Silicon area	Transistor number
Fixed priority	360	6 %	1440
Round-robin priority	580	9.66 %	2320
Control unit	1300	21.66 %	5200
PCI Interface	950	15.83 %	3800
DMA Interface	910	15.16 %	3640
SCSI Interface	830	13.83 %	3320
FIFO Interface	320	5.33 %	1280
DRAM Interface	350	5.83 %	1400
4-phase Interface	400	6.66 %	1600
Controller	6000	100 %	24000

Table 1. Logic level results of the controller.

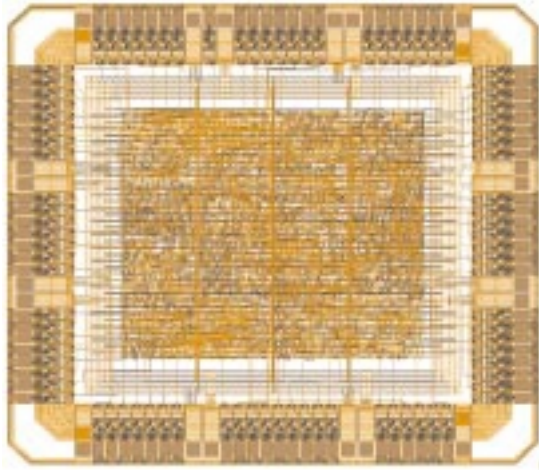


Figure 6. Layout of the design.

Laboratory of Electronics and Micro-Electronics. He obtained his DEA in physics (Option Micro-Electronics) at 1996. His researches interest, communication synthesis for distributed embedded systems in collaboration with the TIMA-CMP Laboratory, I. N. P. Grenoble, France

REFERENCES

1. G. P Hyett, United State Patent, Pat. No. 4942516, July, 1990.
2. VITA. VMEbus specification manual. PRINTEX Publishing, 1985.
3. "IEEE Standard VHDL Language Reference Manual," IEEE, N.Y., 1988.
4. A. Zitouni, C. Souani, M. Abid, K. Torki, R. Tourki, "Communication synthesis approach for distributed systems", Techniques and Sciences Informatiques (TSI), Hermes Science Publication, Avril 2000.
5. S. Narayan and D. Gajski., "Interfacing incompatible protocols using interface process generation", in: Proc. IEEE Design Automat. Conf., June 1995, pp. 468-473.

AD POLYHEDRA