A VLSI PROCESSOR-SWITCH FOR A DUAL IEEE-796 BUS
WITH SHARED AND DUAL-PORT MEMORIES

J. Opsommer and E.H. D'Hollander
State University of Ghent
Department of Electrical Engineering
St.-Pietersnieuwstraat 41
B-9000 Gent, Belgium

Abstract
In many shared memory multiprocessors the limited bus bandwidth is enhanced by a multistage interconnection network or a pipelined memory access. A less costly approach however is to use a multiple bus system. This increases the bandwidth, while preserving the software characteristics of a single bus system through an interleaved memory addressing scheme.

We have developed a VLSI switch which controls and arbitrates the signals of a double Multibus (IEEE-796 standard) system. There is one switch per processor, which is placed between the processor and the buses. Besides the pure shared memory accesses, the switch will also recognize requests for another processor's dual-port memory and route the data transfer accordingly.

The chip was designed with Silvar Lisco CAD software and simulated using the HILO-3 development tools. A detailed simulation of the standard cells was possible using a library of gate level models with the appropriate timing characteristics, delivered by the chip manufacturer. Functional models of the processor and the memories, together with the gate level switch model provided a complete dynamic simulation of the multiprocessor system. The simulation allows a full check of the VLSI component including a functional description of processor and memories, before the manufacturing process.

The timing specifications of the first samples have been verified and currently the switch is being incorporated into an existing multiprocessor prototype. In contrast with theoretical multiple bus simulation, this study has resulted in an accurate performance evaluation which is confirmed by the in-circuit measurements.

I. Introduction

In this paper, a VLSI design of a dual bus switch for the Multibus (IEEE-796 standard) is presented and the results of a detailed gate level simulation are given. The switch will be used in an existing multiprocessor as primary building block for an interleaved memory access, thereby increasing the processor-memory access bandwidth.

Besides an increased bus bandwidth, a multiple bus system offers the possibility of a fault tolerant memory path, because a failure of one bus can be resolved by a graceful degradation. Markovian queuing models [1, 2, 6-8] and detailed simulation models [5] indicate that multiple bus solutions provide a cost-effective alternative with respect to the crossbar switch. However, data which describe the performance of a true multiple bus realization are scarce.

In the following Section, the bus operation is described with respect to a dual bus operation. In Section III the HILO-3 modeling technique and the main simulation components are introduced. Their implementation in an existing multiprocessor is explained in Section IV. The results of the gate level simulation are discussed in Section V. Finally, Section VI contains the concluding remarks and options for future work.

II. Extension of the IEEE-796 Standard

for Dual Bus Operation

Memory Read/Write Cycle

The IEEE-796 bus utilizes an asynchronous handshaking protocol between the master (usually a processor) and the slave (e.g., a memory). Consider for example a write command from the processor to the memory (Fig. 1).

Fig. 1. Memory Write Cycle. 1) write command is activated after address- and data-lines; 2) Memory response time; 3) XACK triggers removal of the write command; 4) Data and address lines are disabled.

Arbitration

In a multi-master system, several processors compete for bus accesses. Since only one processor at a time can use the bus, an arbiter is needed which assigns the bus to one of the competing processors according to an arbitration scheme [4]. A processor i requests the bus by asserting the Bus REQUEST (BREQ) input to the arbiter. The arbiter grants the bus to that processor by the BPRN line. When the bus is or becomes free, indicated by a high BUSY signal, the processor takes the bus (Fig. 2a). We note in passing that all bus signals are active low.

Fig. 2a. Processor 1 is granted the bus.
In a dual bus configuration, this mechanism requires some modifications. The dual bus switch has to route the arbitration signals from the processor to the arbiter of the selected bus. Since the selection of a bus is based on the lower address bits, e.g. an even/odd addressing scheme, these lines must be made available before any bus request is made. However, the standard IEEE-796 protocol asserts the lines BREQ and ADDRESS in reverse order. It follows that the standard bus signals cannot be used and the selection mechanism has to fetch the lower address lines via a separate connection (Fig. 2b).

III. HILO-3 Language Models

HILO-3 Overview

HILO-3 is a simulation language for VLSI design. Its power stems from the concise way with which multiple processes can be created in a single functional model, the hierarchical circuit description and the library support to store the characteristics of the elementary VLSI components. HILO-3 allows a full check of the VLSI component in its future environment, and a performance evaluation before the chip is actually built.

Circuits are defined in a Hardware Description Language (HDL). Two types of models can be combined: structural and functional. A structural model describes a circuit in terms of elementary gate primitives (AND, OR, NOR, ...). In functional models however, the designer explicitly specifies what actions must be performed when a particular event, i.e. a transition in the state of the circuit, occurs.

The definition of an asynchronous process is as follows:

WHEN (event) DO (action list);

The (event) condition triggers the sequence of actions, in the (action list). The NEXT clause specifies the next (event) in the same process (started by a WHEN), for which an (action list) exists, e.g.:

WHEN (event1) DO (action list 1);
NEXT (event2) DO (action list 3);

When an (action list) is conditioned upon several consecutive (events), the combined event is described

by a THEN construct. The triggering can be stopped by a RESET condition:

WHEN (event1) THEN (event2) RESET (event3) -
DO (action list);

The (action list) is executed only if (event1) and (event2) occur in this order, and no (event3) in between.

Processor and Memory Models

In order to simulate the design of the switch, a functional hardware description of the processor, the memory and the arbiter have been written in HDL. In order to get an idea of the modeling language, a synopsis of the memory model is presented in Fig. 3. Two processes are defined, corresponding to the WRITE and the READ command respectively. When a write event occurs (W goes low), the data is copied into the memory array and after a memory cycle time of 360 ns an acknowledge signal XACK is sent. The XACK signal is removed 340 ns after the write command is dropped by the processor. The read process is programmed accordingly.

IV. Implementation of the Switch

in a Multiprocessor Environment

The VPS Multiprocessor

In recent years, the VPS (Virtual Processor System) was designed, built and tested. The VPS is a multiprocessor system consisting of up to 21 boards with a 8086/8087 processor pair, linked by the IEEE-796 bus. All processors share 760Kb of memory and every processor can directly communicate with a 16Kb dual-port memory of a peer processor-board. The multiprocessor is used for the automatic partitioning of high level programs and a
With more or faster processor boards however, we foresee a significant performance drop due to the bus saturation. This can be counteracted by a dual bus system, where each of p processors talks to the shared memory via two buses (Fig. 4). The dual bus switch operates in the following way.

![Diagram of dual bus switch](image)

**Fig. 4.** The VPS Multiprocessor. Dual Bus Configuration

### Switch Operation for Dual-Port Memory Accesses

The inclusion of dual-port memories in the multiprocessor adds some complexity to the switch controller, since in order to eliminate race conditions, two dual-port commands must not occur simultaneously. A race can occur when two processors address each other's dual-port memory using a different bus. Since a processor can only redirect the switch towards the other bus when its own data transfer is finished, both switches and both processors will wait forever for their dual-port access to terminate.

The solution adopted here is to use only one bus for dual-port accesses. Because the dual-port areas occupy the high addresses, a dual-port select signal, DPSEL is generated internally, based on address lines A17-A19. DPSEL overrides SMSEL and always selects bus 1.

When the controller of an idle processor detects a dual-port activity on the bus (signal DPSEL), it

### Switch Operation for Shared Memory Accesses

The main part of the switch is the VLSI controller, built from standard CMOS cells and contained in a 40 pin package. In order to limit the chip area and the number of pins, the switch itself is composed from off-the-shelf fast Schottky TTL components.

In a 1 processor-2 memory configuration, the processor communicates only with the CONTROLLER and the SWITCH. The relevant pin signals of the processor, the memories and the switch are shown in Fig. 5.

![Diagram of switch signals](image)

**Fig. 5.** Major signals of the switch controller and the switch in a dual bus configuration with one processor and two memories.

When the processor issues a bus request BREQ, the controller checks SMSEL, which corresponds to address line 3 and indicates a longword boundary. SMSEL points to the requested memory bank. BREQ is routed to the proper bus arbiter and the bus grant signal BPRN is returned to the processor. When the bus is released by the previous master, the BUSY signal is enabled. Next the data, address and N/W command lines are routed through the switch by activating the signals DBR, AXN and RMN (i=1,2). The direction of the data, DTRAN, depends on the read or write command, while the direction of the address bus signal and the address tag must be forwarded. When the command is complete, the acknowledge signal of memory bank i, XACKi, is routed back to the processor by enabling the proper XACKENi signal. The transfer is terminated when the processor lowers its BUSY signal, at which point the controller disables all signals.

### V. Simulation Results

Two configurations are compared, one with a single and one with a double bus. Both setups consist of two processors A and B and one or two memories respectively. In the subsequent simulated instruction sequence, processor A writes to memory 1 and then to the dual-port memory of processor B, while processor B executes one write to memory 2. The major timing events are represented in Table 1.

In the single bus operation (Fig. 6a), processor A gets the bus first. Processor B has to wait 1035 ns for the bus grant signal (BPRN/) from the bus arbiter, and then waits another 272 ns for the bus to be released by processor A (BPRN/-BUSY). Both memory writes require 2513 ns.

In a dual bus operation (Fig. 6b), processor A and processor B proceed in parallel, each using a different bus. The two write commands now take 1447 ns, including 140 ns switching overhead. This corresponds to a switching overhead of about 10% for a pure shared memory access.
The design and the simulation of the IEEE-796 bus switch, using integrated CAD tools, required only four man months to develop from scratch. With respect to a conventional PCB design we experienced two major advantages. First, the complete project could be prototyped without any hardware effort. The hierarchical graphics support allows a flexible and incremental design, where simple components can be used as building blocks in a larger system. Second, the simulation tools greatly enhance the confidence level with which the chip is built. Using a bottom-up approach, the elementary components are tested separately, and then put together in a global simulation of the chip in its target environment. The simulation results using the estimated gate delays, indicate a switching overhead of about 10%, yielding a bandwidth of 180% with two buses. At this moment, the chip is produced and tested. It's implementation resulted in a small double Eurocard PCB. The first results confirm the simulated performance analysis.

**VI. Conclusion**

The design and the simulation of the IEEE-796 bus switch, using integrated CAD tools, required only four man months to develop from scratch. With respect to a conventional PCB design we experienced two major advantages. First, the complete project could be prototyped without any hardware effort. The hierarchical graphics support allows a flexible and incremental design, where simple components can be used as building blocks in a larger system. Second, the simulation tools greatly enhance the confidence level with which the chip is built. Using a bottom-up approach, the elementary components are tested separately, and then put together in a global simulation of the chip in its target environment. The simulation results using the estimated gate delays, indicate a switching overhead of about 10%, yielding a bandwidth of 180% with two buses. At this moment, the chip is produced and tested. It's implementation resulted in a small double Eurocard PCB. The first results confirm the simulated performance analysis.

**References**


