Design of a Communication and Shared Memory Management Unit

Joaquim Castro Ferreira, António Rui Borges

***Abstract* – In this paper we present an overview of the architecture, the design process and the main properties of a communication and shared memory management unit for a MIMD type multiprocessor mesh.**

The circuit has been completely specified and was later designed as a standard cell ASIC based on ES2[1] ECPD10 library of cells, for a N-Well CMOS $1.0\mu m$ process. The CAD framework used was Cadence Edge (ES2 SOLO 2030).

***Resumo* – Neste artigo apresentam-se as principais caracteríisticas da arquitectura e as etapas de projecto de um circuito gestor de memória partilhada e de comunicações pertencente a uma malha multiprocessador do tipo MIMD.**

Este circuito foi completamente especificado, tendo sido posteriormente implementado sob a forma de um circuito integrado semi-personalisado de aplicação específica (ASIC), baseado em células lógicas padrão.

A biblioteca de células lógicas padrão usada foi a ES2 ECPD10, para a tecnologia N-Well CMOS $1.0\mu m$. A ferramenta de CAD utilizada foi o Cadence Edge na sua configuração ES2 SOLO 2030.

I. INTRODUCTION

This paper describes the development of a communication and shared memory management unit to be integrated in a multi-port memory module belonging to a MIMD[2] type multiprocessor mesh (figure 1), described in [1]. This multiprocessor architecture consists of two types of modules: the processing modules and the multi-port memory modules.

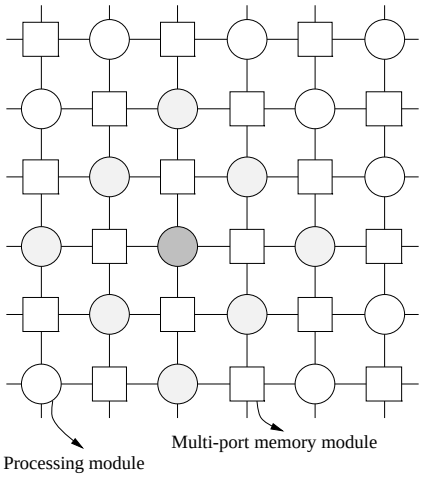

Figure 1 - System architecture.

[1]European Silicon Structures
[2]Multiple Instruction Multiple Data

The function of the multi-port memory modules is dual. On one side they act as a communication channel for message passing between adjacent processing modules[3]; on the other side, they are a location where adjacent processing modules share and process common data.

In the initial prototype [2], only mailbox and access control to the shared memory are hardwired. All the remaining control functions (management of the shared memory and communication channels) are organised in data structures stored in the multi-port memory modules and are manipulated in a mutual exclusion basis by the distributed operating system that runs on the processing modules. This solution is conceptually simple, but it is not very reliable, since any malfunctioning of a processing module can corrupt the data structures.

The present work goals were to develop a new solution on which the control mechanisms of the shared memory and the communication channels between adjacent processing modules, were centralised in a single chip on the multi-port memory modules. This new solution is both more robust and efficient than the former.

II. CHIP ARCHITECTURE AND SPECIFICATION

The block diagram of the new architecture for the multi-port memory module is presented in figure 2.

It is divided in three main blocks, the interface and the communication channels management units, which can be accessed simultaneously by the four adjacent processing modules, and the shared memory management unit, which can only be accessed by a single processing module at a time.

The objectives to be attained can be so summarised:

1. To increase the system reliability by:
   - Centralising in the multi-port memory module all its control mechanisms.
   - Validating every single access either to the shared memory or to the mailboxes.
   - Signalling all the access errors to the responsible processing module.
2. To provide extra functionalities for message passing by:
   - Creating dedicated communication channels between every pair of adjacent processing modules.
   - Devising more extensive mechanisms for mailbox-related interrupt generation.

This new functionalities are described next.

[3]Processing modules that share the same multi-port memory module

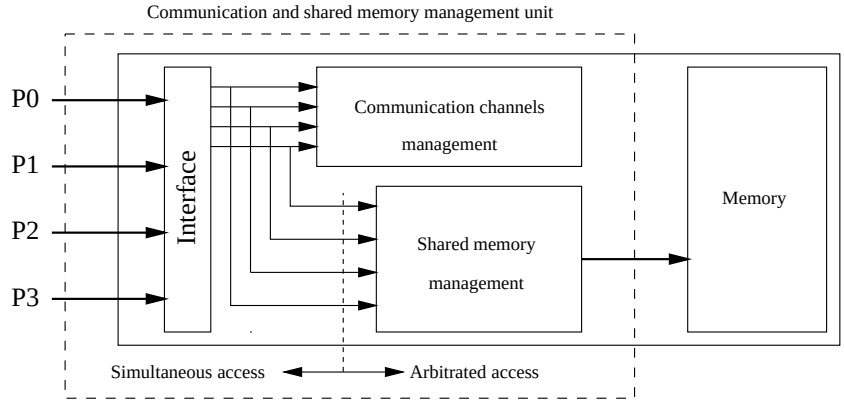

Figure 2 - New multi-port memory module.

A. Shared memory management

The shared memory was divided into 32 equally sized pages, being the first one reserved to address the mailboxes and the internal registers. This particular number was a compromise between system functionality and silicon area needed to build the memory control table.

The objectives of the shared memory management are:

- To provide an extension to the internal memory of the adjacent processing modules.
- To provide functionalities for data transfer between adjacent processing modules.
- To validate every single access to the shared memory.

Allocated memory can be public, when all processing modules have full control of it (including freeing memory pages), or private, when only the processing module that has allocated it can use it. Whenever a processing module performs an allocation operation, memory pages are set to private type. Later on, to share the data stored in them, the processing module has to re-allocate those pages (that is, change their type to public).

According to this, the shared memory management unit has to perform three operations:

1. Allocate memory (private memory).
2. Re-allocate memory (public memory).
3. Release memory.

The status of each shared memory page is centralised in an arbitrated access table. The contents of this table is not visible by the processing modules and has three fields for each page:

- Memory state (free or allocated).
- Type of memory (public or private).
- Identification of the processing module that allocated it.

A set of registers is required to implement the previous operations. The first is the memory control register. A processing module accesses this write-only register after arbitration and sets the number of pages it wants to modify (allocate, re-allocate or free), the base address of the first page it wants to re-allocate or free and the operation code. After checking and modifying the memory status table, the operation result is written to a read-only register (one per processing module), where the corresponding processing module can get the result operation code and, if the case be, the allocated pages base address.

Associated with the shared memory access validation is the invalid access register (one per processing module), which displays the page number where the access error did occur, as well as its type (access to an unallocated page or access to a private page).

B. Communication channels management

The routing of messages circulating in the mesh is performed in a store-and-forward manner [3]. According to this, the communication channels management unit, located in the multi-port memory module, should only be responsible for message passing between adjacent processing modules. The operating system itself is responsible for the global message routing.

The previous scheme of interrupt driven mailboxes was extended in order to increase the available bandwidth for message passing. As it can be seen in figure 3, each single communication channel between every pair of adjacent processing modules has now an unique physical channel connected to it.

To implement such a scheme, 24 mailboxes were needed (because of the two existing levels of messages: system and user).

The message passing protocol is based on the following rules:

- Generate an interrupt on the message receiver, whenever a message is written on one of its mailboxes.
- Generate an interrupt on the message sender, whenever a message is read from one of its mailboxes.
- Generate an interrupt at the appropriate processing module whenever it tries to read from an empty mailbox or write to a full one (invalid access).
- Validate every single access to any mailbox.
- Be able to mask every source of interrupt.

These rules rely on a set of registers to be implemented. Among them, there is one per processing module, the interrupt status register, to store the status of the mailboxes associated with each processing module. By reading the

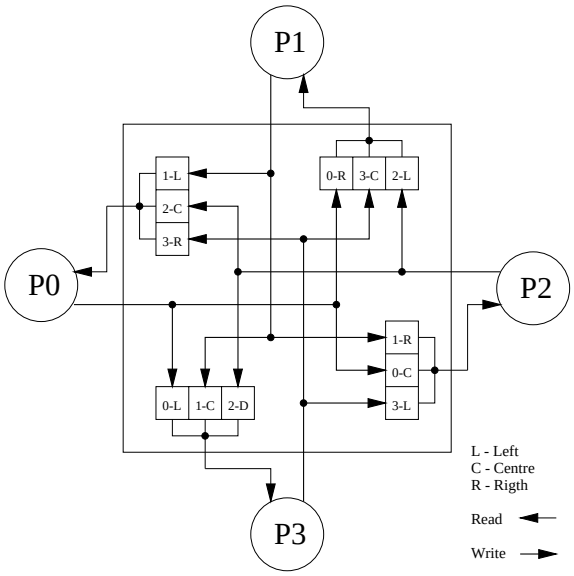

Figure 3 - Organisation of the communication channels associated with the mailboxes.

contents of this register, a processing module can identify the source of interrupt and act accordingly.

Closely related with the interrupt status register is the interrupt mask register, where each processing module can specify the conditions on which it can be interrupted.

A last register needed is one that contains the data related with the invalid access to the mailboxes.

III. CHIP DESIGN

The chip design was divided into the three blocks of figure 2.

The interface unit implements the processing modules bus protocol and decodes the resources mapped in the first page of the shared memory.

The communication management and memory management units were mostly designed separately. Certain common tasks, however, as the interrupt generation and the buses control, were centralised at a single point.

A. Shared memory management unit

This part of the circuit is made up of four main blocks, as it is shown in figure 4.

- Arbiter.
- Memory management block.
- Interface with the dynamic shared memory.
- Lock-flag for topological search.

The circuit responsible for arbitrating the accesses to the shared resources is based on a round-robin algorithm. Since provision for using dynamic memory was considered, the highest priority was assigned to the refresh request signal.

A time-out mechanism was also included in order to terminate an abnormally long data transaction from a processing module whenever a refresh request is pending longer than the maximum allowed time between two refresh cycles.

The memory management block due to its significance will be described in more detail bellow.

The lock-flag is simply a one bit write-once register used during the power-on procedure to determine the system interconnectivity.

B. Memory management circuit

This part of the circuit is responsible for processing the allocation, re-allocation and free operations, as well as for shared memory access validation.

It consists of a 31-entry memory status table, a write-only memory control register, four read-only operations result registers and by a state machine that controls all these blocks plus some extra logic (figure 5).

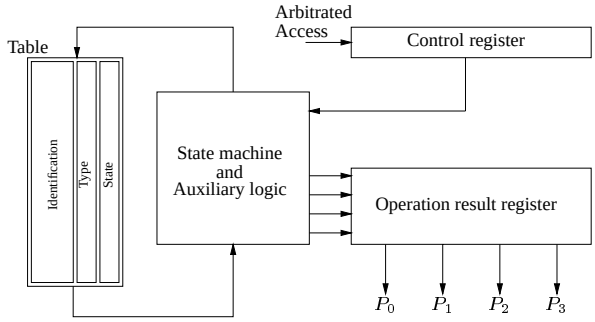

Figure 5 - Functional diagram of the memory management circuit.

Two modes of operation were implemented. In the first one, it processes the operations related with shared memory by checking and modifying the memory status table in a sequential way and allocating memory pages on a first-fit basis. In the second, the memory status table is directly addressed by the processing module to validate the access to a shared memory page according to the information stored in its entry.

C. Communication channels management unit

As it can be seen in figure 6, this unit is made of four main functional blocks:

- Access validation.
- Interrupt status register.
- Interrupt generation.
- Mailboxes.

The access validation guarantees the integrity of the data stored in the mailboxes, by making impossible to write into a full mailbox or to read from an empty one. A mailbox is considered full as soon as a write operation into its last position is performed; on the other hand, it is considered empty as soon as a read operation from the same position is completed. Thus, simultaneous reading and writing into the same mailbox is prevented.

The validation logic uses the data stored in the interrupt status register to decide whether access to the mailboxes should be permitted. In the case of an invalid access an interrupt is generated and a dummy access cycle is performed.

The mailboxes were implemented using Dual-Port RAMs (DPRAMs) available in the library. Each DPRAM holds two mailboxes corresponding to a communication channel between adjacent processing modules. As it can be seen in

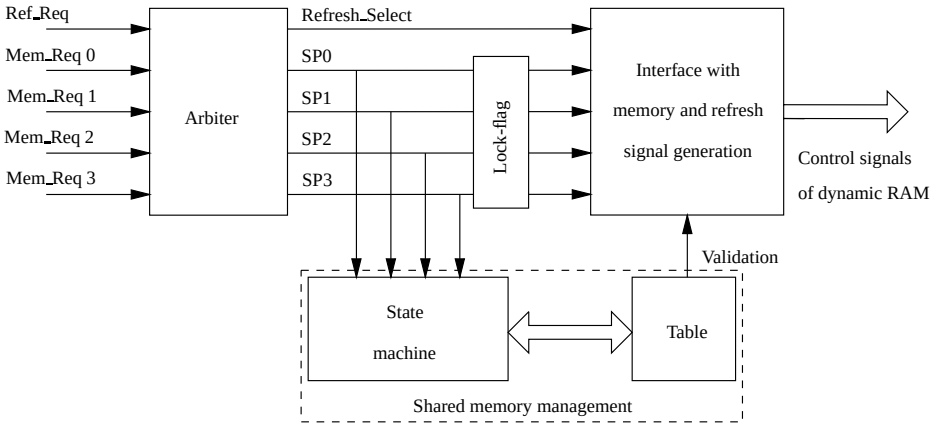

Figure 4 - Shared memory management block diagram.

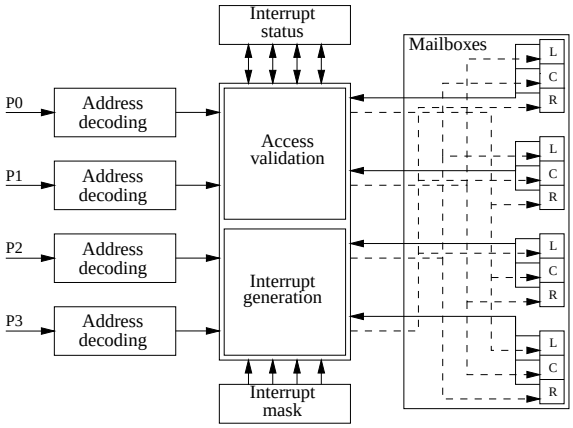

Figure 6 - Functional diagram of the communication channels management block.

figure 7, the DPRAMs were configured in such a way that one port is read-only and the other is write-only.

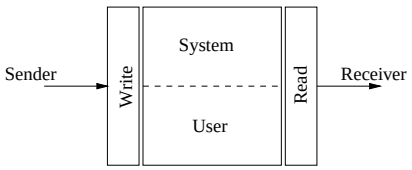

Figure 7 - Implementation of two mailboxes into a single DPRAM.

The alternative option would be using FIFOs. This option, however, would be more demanding in terms of silicon area, since 24 FIFOs would be required instead of the 12 DPRAMs used.

IV. SIMULATION

Circuit simulation was carried out step by step. A bottom up approach was first followed as the different basic logic blocks were designed. The main aim was to spot any logic design error. This was done with more detail for the most complex blocks, as those which included state machines or elaborated interconnections (mailboxes access validation, for instance).

Next, the basic blocks were successively grouped together into a single schematic in a hierarchical manner and the process repeated at each integration level. In the end, the resulting circuit was submitted to a high level functional test. The test vectors set aimed to model the real world operating conditions of the circuit. Since the circuit was divided in two parts, the construction of the test vectors was also divided in two steps.

To begin with all vectors related with the shared memory management were written. An example of this procedure that shows memory occupation over time is presented in figure 8.

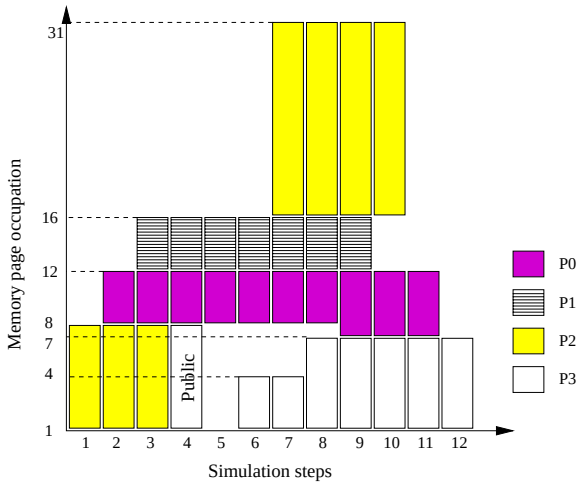

Figure 8 - Evolution of memory occupation along the simulation steps.

Afterwards, the simulation vectors related with the communication channels management were written. The methodology followed in this case was to write and read in every location of the mailboxes address space, keeping at the same time all the interrupt sources enabled.

In the simulation process it was used the SILOS2 simulator (ES2 version of Cadence SILOS) and the test vectors were written using STL language.

V. CHIP PROPERTIES

Once the structural description of the circuit was ready and fully simulated, the placement of the cells and the routing

of the interconnections were carried out.

As it can be seen in figure 9, the circuit is strongly core limited. Most of the silicon area (85%) is occupied by the routing channels. This very high value was somewhat expected giving the number of buses involved. However, we believe that the situation could have been improved if the placement and the routing were controlled and carried out respecting the design hierarchy. Nowadays, the new framework from Cadence, Opus, solves this problem.

The 12 large blocks that appear near the lower left corner of the layout (figure 9) represent the DPRAMs.

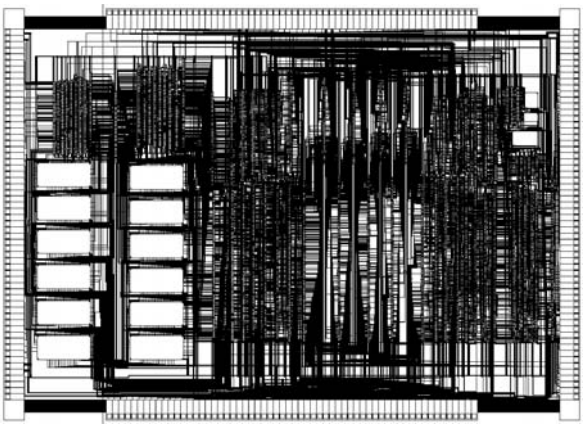

Figure 9 - Final chip layout.

Chip area	105 mm^2
Standard cells total length	147.932 mm
Standard cells area	7.4 % of total chip area
Macro cells total area	7.5 % of total chip area
Number of macro cells	15
Number of I/O cells	240
Number of nodes	6113
Number of interconnections	13315
Number of standard cells used	6267
Number of gates	28020
Number of transistors	112081
Maximum clock frequency	16 MHz

Table I

MAIN CHIP PROPERTIES.

Circuit performance was assessed through simulation. It proved to work successfully at about 16 MHz under worst-case conditions. The operating frequency was mainly limited by the PLAs delay time that were used to implement the memory management state machine. Their poor performance may be one of the reasons why new versions of ES2 libraries for Cadence Opus do not include anymore PLA macro-cells.

VI. CONCLUSIONS

The circuit was completely designed and fully simulated, but it was not sent to fabrication due to its high cost.

Silicon area restrictions have imposed that no post fabrication verification structures were implemented. However, a statistical fault simulation [4], considering 10% of the faults, was carried out using the functional test vectors set. It gave a value of 58.8% for the fault coverage. This result can only be considered in a qualitative way because several cell types are excluded from the fault coverage computation by the fault model implemented by ES2 [5].

REFERENCES

[1] A. R. Borges, *Estruturas Computacionais para Reconstrução de Imagem em Tomografia*, PhD thesis, Universidade de Aveiro, 1994.

[2] A. R. Borges and A. M. B. Ferrari, "A pipelined MIMD architecture for tomographic reconstruction", *ECCV Real-Time Vision Workshop*, April 1990.

[3] C. L. Seitz, E. W. Mayr, W. Dally, S. L. Johnsson, Y. Abu-Mostafa, R. Lyon, and B. Ackland, *VLSI and Parallel Computation*, Morgan Kaufmann Publishers, 1990.

[4] S. A. Al-Arian and M. A. Al-Kharji, "Fault simulation and test generation by sampling techniques", in *International Conference on Computer Design: VLSI in Computer and Processors*. IEEE, 1992, IEEE Computer Society Press.

[5] ES2, *Engineering Application Notes*, European Silicon Structures, 1993.