Suitability of Using Microcontrollers in Implementing new P-System Communications Architectures

Abraham Gutiérrez, Luís Fernández, Fernando Arroyo, Santiago Alonso

Natural Computing Group - Universidad Politécnica de Madrid - Spain
(Tel : +34-91-336-7891; Fax : +34-91-336-7893)
{abraham, setillo, farroyo, salonso}@eui.upm.es

Abstract: The distributed implementation of P-Systems has met with the communications bottleneck problem. When the number of membranes grows in the system, the network gets congested and the times to execute an evolution step degrade.
Several published analysis have proved that there is a very strong relationship between communication time and evolution rules application time in membranes of the system. Moreover, recent works present analysis for distributed architectures that are technology independent, based on: allocation of several membranes in the same processor; the use of proxies for communication among processors; and, token passing in the communication. These solutions avoid communication collisions, and reduce the number and length for communication among membranes. All these facts allow to obtain a better evolution time than in others suggested architectures which they get congested quickly by network collisions when the number of membranes grows.
The aim of this work is to do an extrapolation of the communications architectures denominated "*partially parallel evolution with partially parallel communication*", to analyze their suitability to be implemented using a low cost universal membrane hardware/software component based on microcontrollers.

Keywords: P-System, architecture, communication, microcontroller, distributed

I. INTRODUCTION

Nowadays, membrane systems have been sufficiently characterized from a theoretical point of view. Their computational power has been settled, many variants are computationally complete. However, the way in which these models can be implemented is an open problem today. As usually happens, implementation of these systems has been attacked from two different approaches: software and hardware models. On the other hand, the hardware model seems to be the most appropriate -apart from the biological implementation– in order to obtain the massive parallelization that membrane systems claim.

Several research works on implementation of P-Systems have been focused on the massively parallel character of the model. In particular, it is possible to find out implementation of P-System on cluster of processors, networks of processors, FPGA's ad hoc designed, and microcontrollers. Most of these solutions have been focused, mainly, in the first phase of the P-System evolution describing digital circuit's architectures/designs for the application of evolution rules inside membranes. The phase of membranes communication has not been contemplated or it has simply been performed by shared memory. Authors do not carry out detailed analysis of the importance of the time costs during communication phase in the total time of P-System evolution. Hence distributed implementation of P-System has met with the communications bottleneck problem. When the number of membranes grows up in the system, the network gets congested and the communication time to execute an evolution step degrades.

Recently, several works have presented new communication architectures approaching in the best possible way the inherent features of P-Systems: obtaining a balance between the previous solutions, allowing a certain grade of parallelism in the rules application phase, as well as in the communication phase of a P-System. The goal of this work is to probe the suitability of using microcontrollers in the implementation of these new architectures.

This paper is structured in the following way: in the first place, related works are enumerated analyzing the proposed architectures; next the basic unit circuit is introduced. Afterward a hardware model based on this basic unit is presented stating detailed analysis of the architecture. And finally conclusions are presented.

II. RELATED WORKS

In Syropoulos [1] and Ciobanu [2] distributed P-Systems implementations are presented. They use respectively, the Java Remote Method Invocation (RMI) and the Message Passing Interface (MPI) over a PC cluster's Ethernet network. These authors don't make a detailed analysis about the importance of time in communication phase with respect the total time of P-System evolution, although Ciobanu [2] declares that "*the response time of the program has been acceptable. There are however executions that could take a rather long time due to unexpected network congestion*".

In reply to this problem, Tejedor [3] presents an analysis of an architecture named "*partially parallel evolution with partially parallel communication*". This architecture is based on the following pillars:

a. Membranes distribution: At each processor, K membranes are allocated that will evolve, at worst, sequentially. The physical interconnection of processors is made through a shared communication line. In this scenario, there are two sorts of communications,
   - Internal communications that are the ones that occur between membranes allocated at the same processor, with negligible communication time.
   - External communications occurring between different processors because membranes are placed in different processors.
b. Proxy for processor: Membranes that are in different processors do not communicate directly. They do it by mean of proxies hosted at their respective processor. Proxies are used to communicate processors. A proxy assumes communications among membranes of one processor towards the proxy of another one. In the same way, when information from other proxies is received, it is redistributed to the membranes of the processor using the proxy. Proxies reduce the transmission cost, because don't penalize the small packets transmission with the protocol overhead.
c. Tree topology of processors. The benefit obtained by using a tree topology in the processors interconnection is that the total number of external communications is minimized due to proxies only communicates with their direct ancestor and direct descendants.
d. Token passing in the communications. In order to avoid collision and network congestion, it has been established an order in the communication. The idea is not to have more than one proxy trying to transmit at the same time.

Finally, Bravo [4] [5] proposes a variation that permits to parallelize, up to a certain degree, external communications among nodes. This permits to drastically reduce the evolution time and tending to the mass parallel character of P-Systems. The new architecture distributes the processors in a balanced tree of N levels depth and A processors in amplitude, depending on the number of membranes.

From a logical point of view, each subtree requires a particular physical network to reach parallelism on external communications. This way, the processors of intermediate subtrees need 2 communication interfaces, one for the network of the subtree which is root, and one more for network of the subtree which is a leaf. On the other hand, only one interface is required for the processors in levels 1 and N because they are part of just one subtree.

III. THE BASIC UNIT CIRCUIT

The designed prototype expects to simulate the operation of a membrane, and not the whole system. It makes use of three basic components: a microcontroller that carries out the processing unit features, an EEPROM memory that carries out the storage unit features and a communications bus. For a complete hardware description of this prototype see [6] and for complete software description see [7].

1. Processing Unit. PIC16F88 Microcontroller

The microcontroller stores and runs the program that simulates the behavior of the membrane. The possibility of changing this program, without modifying the associate hardware, provides a great flexibility to the solution allowing the simulation of any kind of membrane system.

The election of PIC (Peripheral Interface Controller) 16F88, is determined by its low cost ($1.90), that allows us to think about solutions with a high grade of potential parallelism and also because of being the half rank microcontroller that provides the minimum necessary requirements: enough processing speed (20Mhz, 200 nanosecond instruction execution), suitable word wide (8bits), possibility of manipulating data with a sufficient precision (8bits) and an easy–to-program instruction set (only 35 single word instructions).

2. Storage Unit: 24LC1025 EEPROM

The memory module defines the compartment that delimits the membrane and it is the container of its main components: the evolution rules and the multisets. The large store capacity of this memory module makes the solution less enclosed than hardware specifically designed solutions, being valid for a bigger range of problems.

The presented solution in this prototype uses a 24LC1025 memory module, also manufactured by Microchip Inc. Technology, that gathers a reduced cost ($4,50) and a great storage capacity (128 Kbytes). It uses a low consumption CMOS technology (3mA in writing process), with a maximum writing cycle of 5ms and is guaranteed for more than a million of write operations. It is also compatible with I^2C protocol, supporting the standard mode of 100 KHz and the fast mode of 400 KHz, which allows a suitable speed for the kind of application that we are developing.

3. Communication Bus: I^2C Bus

The communication bus allows the effective communication among the membranes, facilitating the distribution of the information for the whole system. It is also the responsible for content exchange between the memory module and the microcontroller.

The prototype uses the I^2C bus (Inter Integrated Circuit Bus) of Philips Semiconductors implemented in MSSP module (Master Synchronous Serial Port) of the own microcontroller. I^2C defines a synchronous, bidirectional protocol, of master-slave type that uses a serial bus, formed by two threads, to which several devices can be connected by a very simple hardware. Each wired device to the bus is recognized by an only address that differentiates it of the rest of the wired components. One of the advantages from the usage of this kind of bus is its simplicity which allows us to connect new components, for example, we can simply expand the storage capacity of our solution until 512 Kbytes adding other three memory modules (24LC1025) to the same bus, and they could be accessed making use of the available address lines.

4. Basic Unit Circuit

The following figure shows us the general structure of the circuit used for the implementation of universal membrane hardware component; the connection between the microcontroller and the memory module through the I^2C bus:

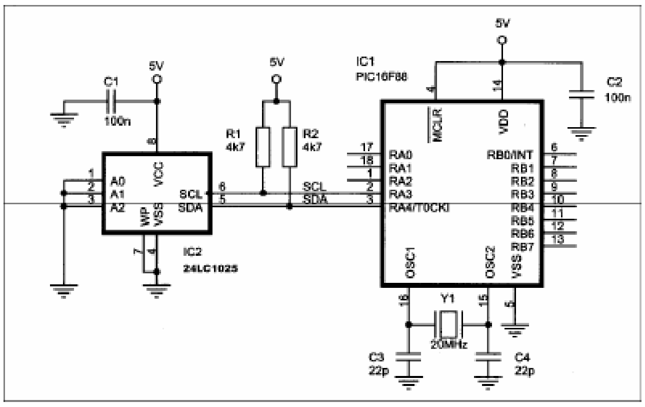

Fig.1. Basic Unit circuit

IV. NETWORK ARCHITECTURE

We try to generate a tree topology like the distributed architectures propose, but adapted to use the basic component presented in the previous point. In the adaptation process appear different problems inherent to the use of microcontrollers and the selected bus of communications.

In order to maintain the productivity of the basic units it is necessary to place communication control processes outside these units. Hence, the selected devices facilitate this tasks distribution.

1. Increase of I^2C basic component interfaces

The microcontroller has already an I^2C port but it needs one or more additional I^2C ports to interface with the other basics units that cannot be located on the same bus that the memory modules (only four memory chips can reside in the same bus).

This problem can be solved with a bus controller that can be used to convert 8-bit parallel data into additional multiple-master capable I^2C port. The bus controller PCA9565 (CMOS integrated circuit from NXP Semiconductors) serves as an interface between the standard parallel-bus microcontrollers and the serial I^2C bus, and allows the parallel bus system to communicate bi-directionally with the I^2C bus. This is commonly referred as master bus. Communication with the I^2C bus is carried out on a byte-wise basis using interrupt or polled handshake and it controls all of the I^2C bus specific sequences, protocol, arbitration, and timing.

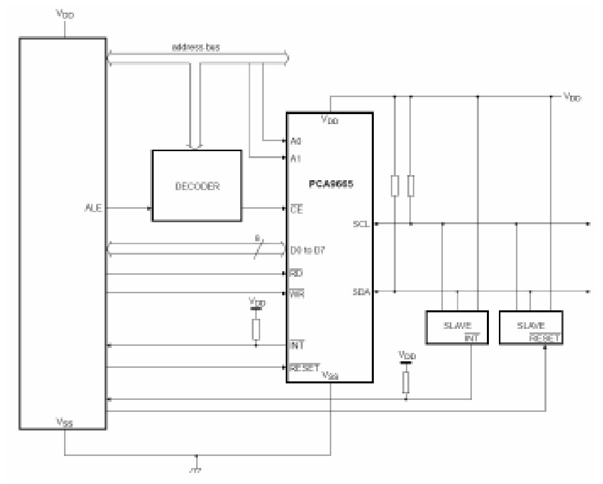

Fig.2. Bus controller connection.

2. I^{2}C Bus Amplification

The I^{2}C reference design has a 7-bit address space with 16 reserved addresses, so a maximum of 112 nodes can communicate on the same bus. Recent revisions of I^{2}C can host more nodes, and also support other extended features, such as 10-bit addressing. So the maximum number of nodes is obviously limited by the address space, and also by the total bus capacitance. The I^{2}C-bus capacitance limit of 400 pF restricts the number of devices and bus length (using shielded cable the maximum capacitance limitation of I^{2}C results in a max line length of about 0,91 meters; so this can be a real problem).

Using the PCA9518A (CMOS integrated circuit from NXP Semiconductors) enables us to divide the bus into five segments off of a hub where any segment to segment transition sees only one repeater delay and no functional degradation of system performance. Any segment of the hub can talk to any other segment of the hub. Bus masters and slaves can be located on all five segments.

While retaining all the operating modes and features of the I^{2}C system, it permits extension of the I^{2}C-bus by buffering both the data (SDA) and the clock (SCL) lines, thus enabling five buses of 400 pF. Typically 20 to 30 devices are the limit for the 400pF maximum capacitance specified by the I^{2}C protocol. So using this hub we can extend the use of the I^{2}C bus to 100 upon 150 devices.

The use of this device has other advantages like: isolating a section of a system that has lost its power supply; addressing selected devices if there are multiple devices in the system with the same I^{2}C address; and it supports arbitration and clock stretching across the repeater. Finally if more than 150 basic units are necessary, using multiple PCA9518A parts, any width hub (in multiples of five) can be implemented using the expansion pins.

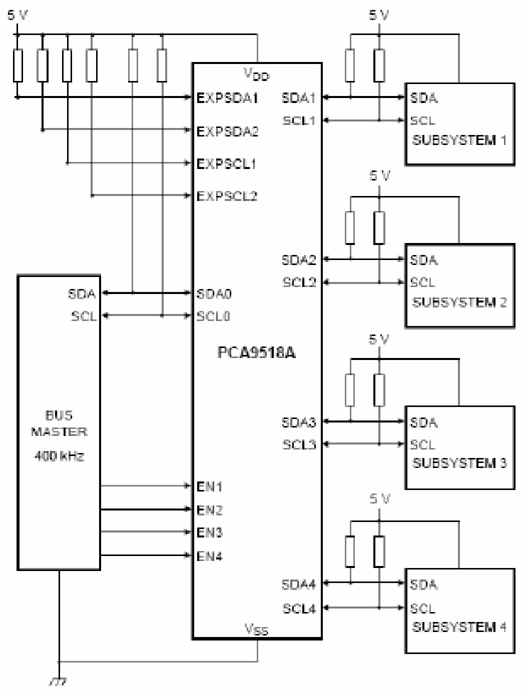

Fig.3. Hub connection.

3. Master-Slave Approach

The I^{2}C bus has two types of nodes, master and slave: a master node is a node that controls the clock; and a slave node is a node that is not in control of the clock line. The I^{2}C bus is a multi-master bus, which means any number of master nodes can be present in the bus. Additionally, a master can also be a slave, and vice-versa.

Overall, there are four distinct modes of operation for a given bus device: master transmit (the node is in control of the clock and is sending data to a slave), master receive (the node is in control of the clock but is receiving data from a slave), slave transmit (the node is not in control of the clock but is sending data to a master) and slave receive (the node is not in control of the clock and is receiving data from the master).

To follow the proposed master-slave architecture, inside every branch we define just one master node. Each slave node houses K membranes, processes multisets and sends the master node multisets whose destination is in a membrane in another slave. The master redistributes the multisets to the proper slaves. The master contains no membranes.

To accelerate the information interchange among the slave nodes, we use a shared memory module. That is used to receive multisets from a slave node or to send multisets to a slave node. All this operations are controlled by the master node. Addresses and data are transferred serially via I^2C-bus.

The selected device is a PCF8598C-2 that is organized as 1024 words of 8-bytes in four 256 word pages. The built-in word address register is incremented automatically after each written or read data byte. All bytes can be read in a single operation. Up to 8 bytes can be written in one operation, reducing the total write time per byte.

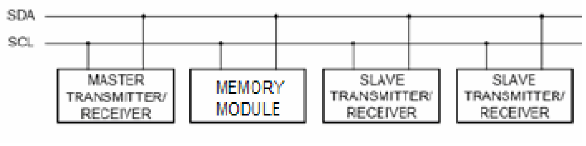
SDA
SCL
MASTER TRANSMITTER/ RECEIVER
MEMORY MODULE
SLAVE TRANSMITTER/ RECEIVER
SLAVE TRANSMITTER/ RECEIVER

Fig.4. Master-slave structure

Every master node monitors the bus for a start bit, and does not start a transmission while the bus is busy. However, two masters may start transmission at about the same time; in this case, arbitration occurs. In contrast to protocols (such as Ethernet) that use random back-off delays before issuing a retry, I^2C has a deterministic arbitration policy. During transmission, each master checks the level of the lines and compares them with the levels it is setting them to.

4. BRIDGE RS232/USB

The capabilities of the hardware solution are expanded upon by using an I2C-USB/RS232 bridge. A desktop with I2C-USB bridge can be used for following tasks: monitoring and debug; acquire and manipulate data from the circuit components; program memories and microcontrollers; train different membrane systems.

The bridge is based on the CY8C24894 Universal Serial Bus (USB) to I^2C adaptor. A serial to I^2C adaptor is also used, for systems with no USB ports.

VI. CONCLUSION

The hardware solution proposed in this paper demonstrates that it is possible implement the communication architectures denominated "*partially parallel evolution with partially parallel communication*", making use of microcontrollers.

The tree topology proposed in these architectures is carried out making use of a PCA9518A hub integrate circuit that enables us to divide the bus into five segments off. To follow the proposed master-slave architecture, inside every branch one master node is defined. These master nodes carry out the features of communications proxy making use of a shared memory to accelerate the information interchange among the different tree nodes.

Finally, to control the operation of the system a RS232/USB bridge of communications has been implemented. Allowing change the membrane system settings from a peripheral unit like a PC and expanding the possibilities of communications with other heterogeneous systems.

REFERENCES

[1] G.Ciobanu, W.Guo. P-Systems Running on a Cluster of Computers. Workshop on Membrane Computing (Gh. Păun, G. Rozenberg, A. Salomaa Eds.), LNCS 2933, Springer, 123-139, 2004.
[2] A. Syropoulos, E.G. Mamatas, P.C. Allilomes, et al, A distributed simulation of P-Systems, A. Alhazov, C. Martin-Vide and Gh. Păun (Editors): Preproceedings of the Workshop on Membrane Computing; Tarragona, July 17-22 2003, 455-460.
[3] A. Tejedor, L. Fernandez, F. Arroyo, et al, Architecture for attacking the bottleneck communication in P-Systems. In: M. Sugisaka, H. Tanaka (eds.), Proceedings of the 12th Int. Symposium on Artificial Life and Robotics, Jan 25-27, 2007, Beppu, Oita, Japan, 500-505.
[4] G. Bravo, L. Fernández, F. Arroyo, et al, A hierarchical architecture with parallel communication for implementing P-Systems. ITA 2007 X-th Joint International Scientific Events on Informatics, June 2007, Varna, Bulgaria.
[5] G. Bravo, L. Fernández, F. Arroyo, et al, Master-Slave Distributed Architecture for Membrane Systems Implementation. 8th WSEAS Int. Conf. on Evolutionary Computing (EC'07) June, 2007, Vancouver (Canada).
[6] A. Gutiérrez, L. Fernández, F. Arroyo, et al. Design of a hardware architecture based on microcontrollers for the implementation of membrane systems, SYNASC 2006, 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing. September 2006, Timisoara, Romania.
[7] A. Gutiérrez, L. Fernández, F. Arroyo, et al. Hardware and Software Architecture for Implementing Membrane Systems: A case of study to Transition P-Systems, DNA13 2007, 13th International Meeting on DNA Computing Memphis, EEUU. June 4-8, 2007.