Hardware Circuit for the Application of Evolution Rules in a Transition P-System

Víctor Martínez, Santiago Alonso, Abraham Gutiérrez

Natural Computing Group - Universidad Politécnica de Madrid – Madrid
(Tel : +34-91-336-7901; Fax : +34-91-336-7893)
({victormh, salonso, abraham}@eui.upm.es)

Abstract: P systems or Membrane Computing are a type of systems based on biological membranes. Transition P systems perform computation through transition between two consecutive configurations. One transition is obtained by applying the evolution rules, which are in each region of the system in a non-deterministic maximally parallel manner. This paper is part of an investigation line which objective is to implement a hardware system that evolves as it does a transition P system. To achieve this objective, it has been carried out a division of this generic system in several stages. The first stage was to determine active rules in a determined configuration for the membrane. The second stage is developed by obtaining the part of the system that is in charge of the application of the active rules. In fact, the circuit obtained in this article counts the number of times that the active rules are applied. In first place, the initial specifications are defined in order to outline the synthesis of the circuit of active rules application. Later on, the design and synthesis of the circuit will be shown, as well as the operation tests, required to present the experimental results obtained.

Keywords: Bioinformatics, Membrane Computing, Robotics, Digital Systems.

I. INTRODUCTION

Membrane Computing or P Systems are computation systems based on the biomolecular processes of living cells [1]. These systems perform a computation through transition between two consecutive configurations by using evolution rules present in each region. If the system reaches a configuration in which there are no applicable rules at any membrane, it is said that the system reaches a halting configuration and, hence, the computation is successful.

This article is a part of a project to obtain a hardware system able to simulate P systems evolution. This generic system has been divided into several stages. The first stage develops a FPGA circuit able to determine active rules in a determined configuration for the membrane [2] y [3]. In this document, the second stage is developed: a circuit for the application of the active rules obtained in the first stage. In general, the application of the active evolution rules for a region of an evolution P system is a repetitive process that may be implemented following different algorithms. A previous study [4] of the possible positive and negative aspects each one of these application methods, allows us to obtain different circuits or systems that differ complexity level and expense of resources.

The first process to obtain the rules that can be used in a certain evolution step consists on obtaining the active rules. From the active rules, we select in a non-deterministic manner, all the rules that will be applied in parallel in each one of the regions. The rules also proceed in parallel to the transformation of the regions contents and of the own structure of the system sending objects and, in their case, dissolving membranes. After the communication and modification of the structure of membranes of P system has taken place, the system has evolved from a configuration to another. The process of application evolution rules in a membrane region is illustrated in the following sequence of steps:

1. Once the system is initialized, the process starts up with the loading of the active rules register *R* obtained by means of the selection active rules circuit.
2. In each iteration, one of the active rules r_i will be applied. This rule is aleatorily obtained.
3. The application of the selected rule r_i consists on the subtraction for the initial multiset ω, of the elements values from the rule antecedent. In turn, we will increase 1 time the particular accountant that counts the number of times that the rule has been applied.

4. With the new multiset ω' obtained it is loaded again in the register *R*, the new set of active rules. Every time that the group of active rules is upgraded, the finish of the process is controlled. The stop condition is obtained when the number of active rules is zero. Therefore, while the cardinal of *R* is bigger than zero, it executes a new iteration of the application process again.

II. HARDWARE CIRCUIT FOR APPLICATION RULES

A first option to obtain the application of the active rules consists on the use of an iterative algorithm of application of the active rules group. In each step of the algorithm, one active rule is choosing in an aleatory way, until draining all the possible applications. This process "*step by step*" may be considered as the most immediate and easy to implement, and the Hardware Circuit for Application Rules presented implements the different processes from this algorithm. The ultimate circuit is obtained from the assembly of the different Functional Units created, along with the control sequential logic. The sequential logic determines the evolution of the internal steps that must take place inside the system until reaching the condition of shutdown. This condition will occur when the active rules register is empty. On a simplified way, the main functional units will be those that implement each one of the steps of the used algorithm, this is:

1. To obtain the active rules: $R \leftarrow ActiveRule$
2. To choose a single rule randomly: $r_i \leftarrow Aleatory(R)$
3. To update the objects multiset: $\omega' \leftarrow (\omega - (Antecedent(r_i)))$
4. To count the rules implicated: $counts(1, r_i)$

The functional unit that obtains the active rules consists of a designed, developed and displayed circuit in [3]. In order to update the objects multiset ω, it must be decreased as many times as the values of the antecedent objects of the rule that is being applied indicate. For it, we address the memory with the position of the rule to be applied, and a functional unit of multisets subtraction will obtain the value of the new updated multiset ω. The subtraction multisets functional unit must make the subtraction of each pair of elements mi from each one of the registers that represent the multiset values.

The sequential controller circuit is in charge of sequential activation of different units functional, as well as control of condition of shutdown, that determines the evolution of the internal steps that must cross the system. The sequence of events that must activate the sequential controller generates 4 states. The account continues of cyclical form until the condition of shutdown is reached, that will deactivate the accountant. In the state 0 is loaded objects multiset register ω present, and is calculated the active rules. In the state 1 is loaded *ActiveRules* register and obtained one active rule randomly. In the state 2 is loaded *1 Active Aleatory Rule* register, and is calculated the values of the new objects multiset and the increase of the accountant of applied rules. Finally, in the state 3 is loaded ω' and *AppliedRule* registers.

The operation of the obtained circuit is based on the following process: once the circuit is initialized, with the load of the initial objects multiset register and the ROM memory with the evolution rules, the circuit operation under the supervision of the sequential controller begins, which will cross the different states from the system.

The Fig. 1 shows the detailed scheme of the designed circuit for the rules application. The condition of shutdown of the circuit by means of a function AND that detects when there is no rule to apply; and in this case, disables the sequential controller. The size of the different elements: registers, memory, connections, etc. it has been chosen based on the characteristics of the practical case that will be developed.

III. EXPERIMENTAL RESULTS

Next we will present the complete process of obtaining the rules to apply on a concrete example of a transition P-system region. This will allow us to illustrate the operation of the developed model, as well as to show the tests to make. The definition, according to formal annotation, of the membrane system used is in the expressions and diagram with its structure of regions:

$\Pi = (V, \mu, \omega_1, \ldots \omega_4, (R_1, \rho_1), \ldots, (R_4, \rho_4), 4)$

$V = \{a,b,c,d,e\}$

$\mu = [_1[_2[_3]_3]_2[_4]_4]_1$

$\omega_1 = aac$

$R_1 = \{r_1: c \rightarrow (c,in_4), r_2: c \rightarrow (b,in_4), r_3: a \rightarrow (a,in_2)b, r_4: dd \rightarrow (a,in_4)\}$

$\rho_1 = \{ r_1 > r_3, \quad r_2 > r_3\}$

The circuit to be carried out obtains the region 1 rules to apply. The region 1 input values will be the objects multiset ω_1, the group of rules R_1, the priority relationships among rules ρ_1 and the inner adjacent regions number (two regions in this case). The number of objects V is 5 with decimal multiplicity (0 - 9), therefore, we will need 4 bits to represents each object. And so, a word representing the multiplicity of the 5 objects will occupy 20 bits. The rules group is formed by 4 rules which will be stored in the device memory. Each of the 4 rules will be coded with 64 bits. This is, 20 bits for the antecedent, 20 bits for each consequent of the two inner interior regions, plus 4 remaining bits used to code the priorities mask of each rule with regard to the other ones. The rules are stored, therefore, in a ROM 4x64 bits memory [3].

Like previous step, is due to initialize the system with the load of the initial objects multiset register: *w_a[3..0] = 2(decimal), w_c[3..0] = 1(decimal), w_b, w_d, w_e = 0;* the inner regions existence indication bits *In1 = 1 (Region 1 Exists), In2 = 1 (Region 2 Exists)* and ROM memory with the evolution rules.

Next they are described with detail the states which the system goes through until completing the first cycle of sequence of states:

Cycle 1 of sequence of states: In state 0 the registry ω_i with the present objects multiset is loaded and the process of calculation of active rules begins. In state 1 the active rules register a_1, a_2, a_3, a_4=(1,1,0,0) is validated, and the calculation of *1 Aleatory Active Rule* is qualified. State 2 been validate the load of the *1 Random Active Rule* register r_1, r_2, r_3, r_4=(1,0,0,0) for example, select the information in memory of the rule that allows to update the values of the present multiset $\omega' \leftarrow (\omega - (Antecedent(r_0)))$, and it is increased the accountant of applied rules. State 3 the load of the registry $\omega' = aac - c = aa$ is validated, and the registry applied rules: ar_1, ar_2, ar_3, ar_4=(1,0,0,0).

Cycle 2 of sequence of states: In state 0 the registry ω_i with the present objects multiset is loaded *w_a = 2, w_b, w_c, w_d, w_e = 0;* and the process of calculation of active rules begins. In state 1 is validated the active rules register a_1, a_2, a_3, a_4=(0,0,1,0) and the calculation of *1 Aleatory Active Rule* is qualified. State 2 the load of the *1 Random Active Rule* register r_1, r_2, r_3, r_4=(0,0,1,0) is validated. State 3 is validated the load of the registry $\omega' = aa - a = a$, and the registry applied rules: ar_1,ar_2,ar_3,ar_4=(1,0,1,0).

Cycle 3 of sequence of states: In state 0 the registry ω_i with the present objects multiset is loaded *w_a = 1, w_b, w_c, w_d, w_e = 0;* and the process of calculation of active rules begins. In state 1 the active rules register a_1, a_2, a_3, a_4=(0,0,1,0) is validated, and the calculation of *1 Aleatory Active Rule* is qualified. State 2 the load of the *1 Random Active Rule* register r_1, r_2, r_3, r_4=(0,0,1,0) is validated. State 3 are validated the load of the registry $\omega' = a - a = 0$ and the registry applied rules: ar_1,ar_2,ar_3,ar_4=(1,0,2,0).

As we can verify, after 3 complete system evolution cycles, the process stops because the condition of shutdown is reached. The obtained result corresponds with the awaited one: rule 1 must be applied 1 time, and rule 3 must be applied 2 times. The Fig. 2 shows the chronogram that illustrates the sequence of states of the system.

IV. FIGURES

Fig.1. Scheme of the designed circuit.

Fig. 2. Chronogram that illustrates the sequence of states of the system

V. CONCLUSION

This article presents a way to obtain a circuit capable to obtain the rules inside the P-system membrane that must be applied. The operation and verification of the obtained circuit is shown with a practical case of membrane system. We can verify like, from an initial P system configuration, and based on the external conditions, a satisfactory final result is reached.

The synchronization between the different functional units that compose the system is controlled by the sequential controller. It will be necessary to fit the clock cycles so that the circuit reaches the stable results within each cycle. The obtained circuit behavior is based on the evolution rules stored in memory and the inputs, which correspond with the values of the region state. If the conditions of the region change, the circuit modifies its outputs being adjusted to the new values. This feature is of a supreme importance in order to integrate this circuit as a module that works cooperatively together with other circuits.

This circuit comprises of a complete system that allows implementing the operation and evolution of a transition P system. The following step in this line will be to obtain the way of communication of objects between regions, which allows building each evolution step.

REFERENCES

[1] Gh. Păun, Computing with Membranes, Journal of Computer and System Sciences, 61 (2000), 108-143, and Turku Center of Computer Science-TUCS Report No 208, 1998.
[2] Víctor J. Martínez, L.Fernández, F.Arroyo, A. Gutiérrez. A Hw Circuit for the Application of Active Rules in a Transition P-System Region. Fourth International Conference Information Research and Applications I.TECH 2006. Varna(Bulgaria). Del 20 al 25 de Junio de 2006.
[3] Víctor Martínez, Abraham Gutiérrez, Luis Fernando de Mingo. Circuit FPGA for Active Rules Selection in a Transition P System Region. ICONIP 2008 - 15th International Conference on Neural Information Processing of the Asia-Pacific Neural Network Assembly. Auckland, New Zealand November 25-28, 2008 Lecture Notes in Computer Science ICONIP 2008, Part II, LNCS 5507
[4] L.Fernández, F.Arroyo, J.Castellanos, J.A.Tejedor, I.García, New Algorithms for Application of Evolution Rules based on Applicability Benchmarks, BIOCOMP06 International Conference on Bioinformatics and Computational Biology, Las Vegas (USA), july, 2006.

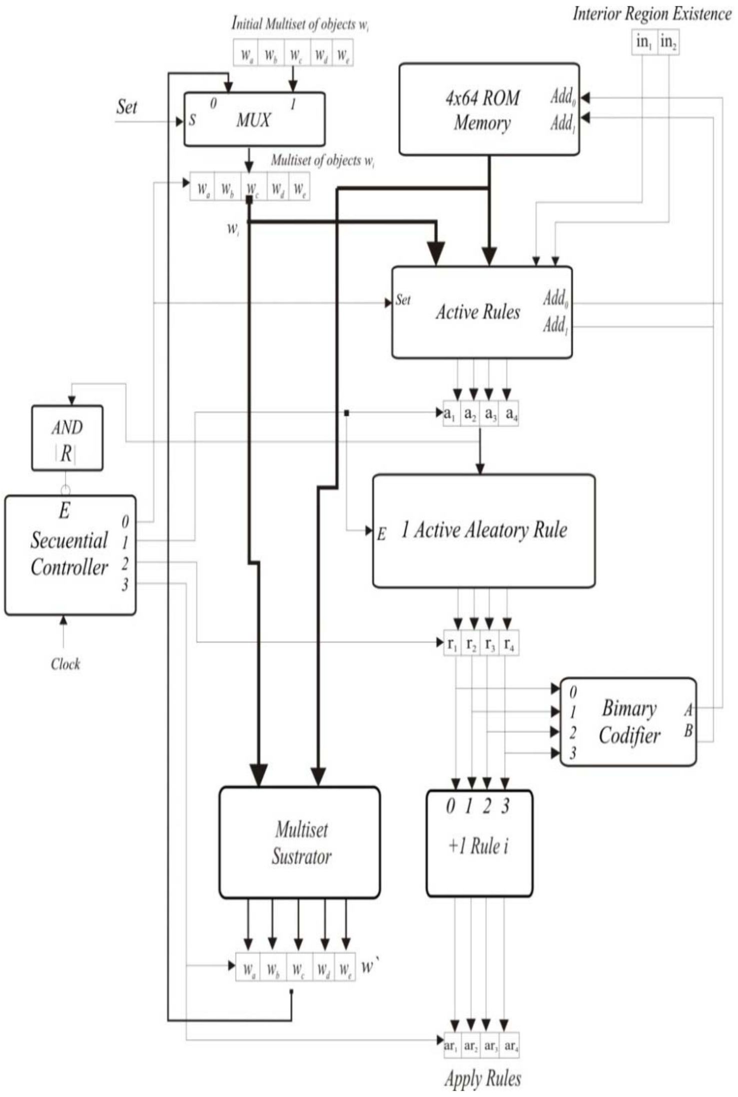

Fig. 1

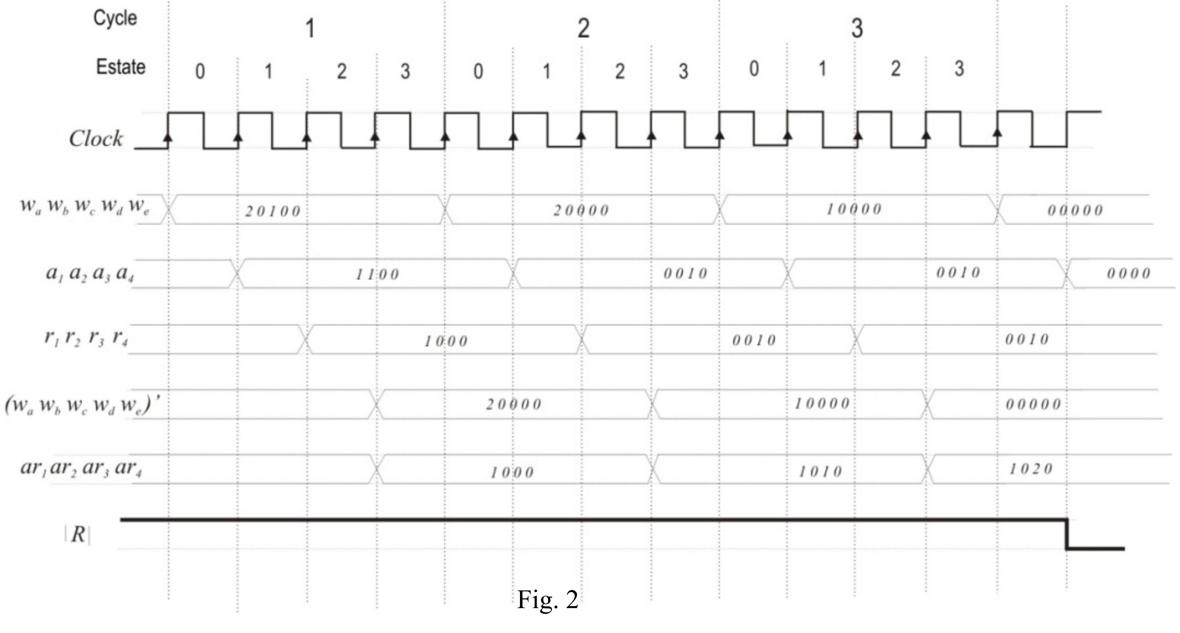

Fig. 2