BUILDING SELF-TIMED SEQUENCE GENERATORS USING SER AND SMER GRAPH DYNAMICS

Ricardo R. Hoffmann ⋆, *Felipe M. G. França* ⋆ *, *Luiz M. C. Branco* ⋆ *and Edson do Prado Granja* †

⋆ COPPE - Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
Caixa Postal 68511, 21941-972, Rio de Janeiro, RJ, Brazil
Emails: {hoffmann,felipe,maltar}@cos.ufrj.br

† Departamento de Engenharia Eletrônica - Escola Politécnica
Universidade Federal do Rio de Janeiro
Caixa Postal 68564, 21941-972, Rio de Janeiro, RJ, Brazil
Email: egranja@lpc.ufrj.br

ABSTRACT

A novel way of building predictable and self-timed sequence generators based on both SER (*Scheduling by Edge Reversal*) and SMER (*Scheduling by Multiple Edge Reversal*) graph dynamics is presented in this paper. In order to illustrate the new methodology, generalized (n bits) SER/SMER-based architectures for Ring counter, Non-decreasing 2^n Module counter and Gray counter are offered. Finally, it is shown how to synthesize the resulting SER/SMER-based self-timed counters using traditional logic gates.

1. INTRODUCTION

One of the main concerns in the development of large-scale asynchronous digital circuits lies on the need for new (asynchronous) cell libraries in conjunction with new CAE tools. This paper introduces a new way of building traditional counters in a self-timed style without the need for specialized asynchronous gates/cells. One expects that this may turn into the main motivation for the development of other self-timed digital structures traditionally used in the implementation of clocked circuits.

The new methodology is solely based on a simple distributed algorithm known as SMER (*Scheduling by Multiple Edge Reversal*) [1][2], a generalization of the SER (*Scheduling by Edge Reversal*) graph dynamics [3]. Both algorithms were proposed in the scope of atomic resource sharing in distributed systems. Dijkstras *Dining Philosophers* problem [4] raised important issues such as concurrency, deadlock, fairness, liveness, mutual exclusion and

*Supported by CNPq, Brazil.

starvation in systems composed by multiple active intercommunicating elements. The design and implementation, using traditional logic gates, of generalized n-bits architectures for SER-based (Ring counter) and SMER-based classic counters, such as Non-decreasing 2^n Module counter and Gray counter, are introduced here.

The remainder of this paper is organized as follows. Section 2 describes the SER distributed algorithm and shows the mapping of a Ring counter into a SER dynamics. Section 3 describes the SMER graph dynamics and the conception of SMER-based versions of the Non-decreasing 2^n Module counter and of the Gray counter. The implementation of the resulting counters using conventional logic gates, in both TTL and FPGA technologies, are described in Section 4. Section 5 presents our conclusions and future works.

2. SER - SCHEDULING BY EDGE REVERSAL

Consider a neighborhood-constrained system composed of a set of *processing elements* (PEs) and a set of *atomic shared resources* represented by a connected graph $G = (N, E)$ where N is the set of PEs, and E the set of edges defining the access topology in the following way: an edge exists between any two nodes if and only if the two corresponding PEs share at least one atomic resource. SER works in the following way: starting from any acyclic orientation ω on G there is at least one *sink* node, i.e., a node that has all its edges directed to itself; all sink nodes are allowed to operate while other nodes remain idle.

This obviously ensures mutual exclusion at any access made to shared resources by sink nodes. After operation a sink node will reverse the orientation of its edges, becoming a *source* and thus releasing the access to resources to its neighbors. A new acyclic orientation is defined and the whole process is then repeated for the new set of sinks. Let $\omega' = g(\omega)$ denote this greedy operation. SER can be regarded as the endless repetition of the application of $g(\omega)$ upon G. Assuming that G is finite, it is easy to see that eventually a set acyclic orientations will be repeated defining a period of length p. This simple dynamics ensures that no deadlocks or starvation will ever occur since at every acyclic orientation there is at least one sink, i.e., one node allowed to operate. Also, it is proved that inside any period, every node operates exactly m times [5].

SER is a fully distributed graph dynamics in which the sense of time is defined by its own operation, i.e., the synchronous behavior is found at the particular case where every node in G takes an exact same time to operate and also a same amount of time to reverse edges. Although the synchronous style will be used ahead to illustrate the functioning of both SER and SMER graph dynamics, asynchronous operation (self-timed in this case), without loss of generality, is an important and interesting property of both distributed algorithms. Another interesting observation to be made here is that any G topology will have its own set of possible SER dynamics [6]. Figure 1 illustrates the SER dynamics.

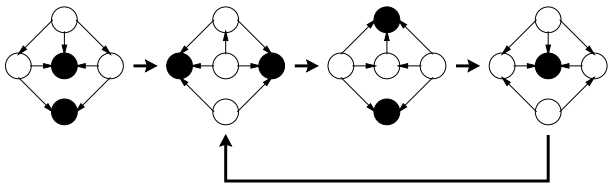

Fig. 1. SER dynamics.

Consider a set of n PEs $P = \{p_1, ...p_n\}$ on which each PE shares a resource with its previous and subsequent PE. This way, PE p_2 shares a resource with PE p_1 and another resource with PE p_3, PE p_7 shares a resource with PE p_6 and another resource with PE p_8, PE p_n shares a resource with PE p_{n-1} and another resource with PE p_1, forming a ring. By setting the orientation of all edges from bigger to smaller PE indexes, an acyclically directed graph will be guaranteed. Figure 2 shows this orientation.

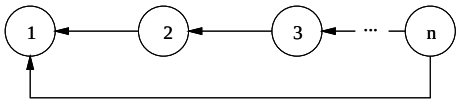

Fig. 2. Ring orientation.

An immediate observation is that a Ring counter behavior is obtained by applying SER to the graph describe above. Figure 3 illustrates a Ring counter for $n = 6$.

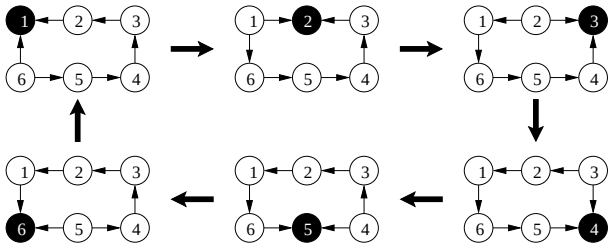

Fig. 3. Six bit Ring counter.

3. SMER - SCHEDULING BY MULTIPLE EDGE REVERSAL

SMER is a generalization of SER in which pre-specified access rates to atomic resources are imposed to processes in a distributed resource-sharing system represented by a multigraph $M(N, E)$. Differently from SER, in the SMER dynamics a number of oriented edges can exist between any two nodes. Between any two nodes i and j, $i,j \in N$, there can exist e_{ij} undirected edges, $e_{ij} \geq 0$. The reversibility of node i is r_i, i.e., the number of edges that shall

be reversed by i towards each of its neighboring nodes, indiscriminately, at the end of operation. Node i is a *r-sink* if it has at least r_i edges directed to itself from each of its neighbors. Each r-sink node i operates by reversing r_i edges towards its neighbors; a new set of r-sinks will operate and so on. Similarly to sinks under SER, only r-sink nodes are allowed to operate under SMER. It is easy to see that with SMER, nodes may be allowed to operate more than once consecutively.

The following lemma states a basic topology constraint towards the definition of M, where *gcd* means the greatest common divisor and f_{ij} is the sum of the greatest multiple of $gcd(r_i, r_j)$ that does not exceed the number of edges oriented from n_i to n_j, and, respectively, from n_j to n_i, in the initial orientation.

Lemma *Let nodes i and j be two neighbors in M. If no deadlock arises for any initial orientation of the edges between i and j, then $max\{r_i, r_j\} \leq e_{ij} \leq r_i + r_j - 1$ and $f_{ij} = r_i + r_j - gcd(r_i, r_j)$.* [2][1]

Figure 4 illustrates the SMER dynamics mimicking the evolution in time of a Non-decreasing 2^2 Module binary counter. The values inside each node on the leftmost graph represent the reversibility (r) of each node. Outside each node there is a name that identify it. Nodes A and B are called *main nodes*, while nodes A* and B* are called *mirror nodes*. Oriented edges between any two nodes are represented by *abacus beads* (or *tokens*) which are positioned according to the desired orientation of the corresponding oriented edges.

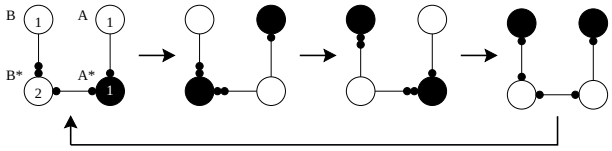

Fig. 4. SMER dynamics.

Consider the binary value 1 for all r-sink nodes and the binary value 0 for non r-sink nodes on Figure 4. Analyzing nodes B and A, they are acting as a Non-decreasing 2^2 Module binary counter, with values 00, 01, 10 and 11 respectively.

For the n-bit generalization of the Non-decreasing 2^n Module binary counter, n main nodes exists, with value 1 for its reversibility (this nodes represents the n bits of the counter). For each of this n main nodes, exists a mirror node linked, whose reversibility depends on which bit of the counter this mirror node represents. The reversibility of the mirror node has the value 2^{i-1}, where i represents the position of the main node in which the mirror node represents. So, for the first bit (the least significant bit - LSB), whose position has value 1, its mirror node will have a value 1 for its reversibility. For the second bit, whose position has value 2, its mirror node will have a value 2 for its reversibility. For the third bit, whose position has value 3, its mirror node will have a value 4 for its

reversibility and so on.

The following is how to set a proper initial orientation of edges. Initially, all edges between the main nodes and mirror nodes are oriented towards the mirror nodes. Besides, all mirrors nodes are linked to the least significant mirror node (LSB), and half of edges in each of this links are oriented to the LSB mirror node. Figure 5a and Figure 5b shows the edges orientation for the Non-decreasing 2^3 Module binary counter and Non-decreasing 2^4 Module binary counter respectively.

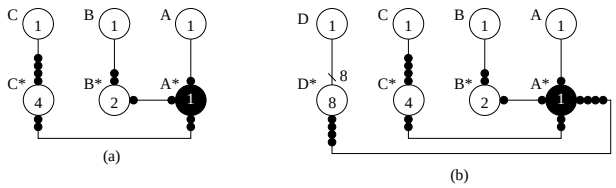

Fig. 5. Edges orientation for the Non-decreasing 2^3 and 2^4 Module binary counter.

We left to the reader the exercise of evolving the SMER counting dynamics on both graphs of Figure 5. Now lets see how to construct Gray counters using SMER. Figure 6 illustrates a two bits Gray counter using SMER.

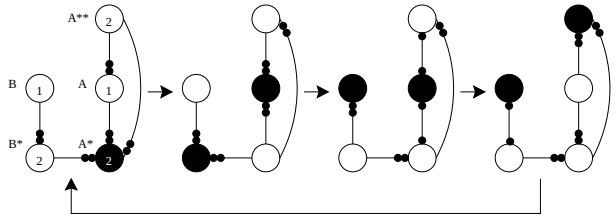

Fig. 6. Two bits Gray counter.

Consider the binary value 1 for all r-sink nodes and the binary value 0 for non r-sink nodes on Figure 6. Analyzing nodes B and A, they are acting as a two bit Gray counter.

For the n-bit generalization of the Gray counter, $n > 2$, n main nodes exists, with value 1 for its reversibility (this nodes represents the n bits of the counter). For each of the n main nodes, there is a mirror node linked (except the LSB main node which has two main nodes linked), whose reversibility depends on the number of bits of the counter.

The reversibility of each mirror node is calculated based on the number of repetitions of bits 1 in a Gray code sequence for each main node. Figure 7 shows the edges orientation for a three bit Gray counter where A, B and C are the main nodes. So, the reversibility of LSB mirror node (A*) is 2. For the second mirror node (B*) is 4 and for the third mirror node (C*) is 4.

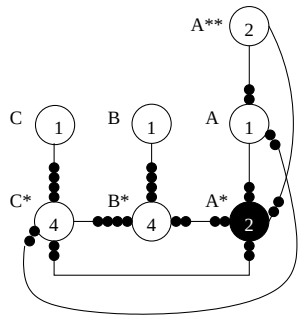

Fig. 7. Edges orientation for a three bit Gray counter.

The following is how to set a proper initial orientation of edges. Initially, all edges between the main nodes and mirror nodes are oriented to the mirror nodes. Note that LSB main node has two mirror nodes (A* and A**) and the edges between main node A and mirror node A** are oriented to A. All edges between A* and A** are oriented to A*. Besides, all mirrors nodes (except LSB) are linked to its right neighbor mirror node and the edges between them are oriented as follows: consider Q* a right neighbor mirror node of mirror node P*; r_p is the reversibility of a mirror node P* and r_q is the reversibility of mirror node Q*; between P* and Q*, r_q edges are oriented to Q* and the remaining edges are oriented to P*.

For $3 \geq i \geq n$, every mirror node i^* is linked to LSB nodes A and A* and the number of edges oriented to LSB nodes is based on the numbers of bits 1 of LSB main node until the main node i changes its value from 0 to 1. Considering the binary value 1 for all r-sink nodes and the binary value 0 for non r-sink nodes on Figure 7 and running the SMER dynamics on this multigraph, the sequence of a three bits Gray counter will be reproduced.

4. DIGITAL CIRCUIT IMPLEMENTATION

The implementation of asynchronous self-timed digital circuits based on the SER dynamics has been explored in [7][8]. Only two basic circuits, namely the *Node Controller* and the *Edge Controller*, are needed.

Node Controllers simply evaluate the presence of oriented edges and, after detecting that the node has become a sink, they allow the target local circuit to operate. After operating for t_{op} time units, the processing logic must produce a corresponding *end-of-operation* signal. Operation also depends on a *Stop/Run* signal. During initialization, while edges are being oriented with proper directions, *Stop* is used. Run is active thereafter.

Edge Controllers connect two nodes using a toggle flip-flop with preset as a edge orientation generator. The preset input is used for initialization, while the output values represent the oriented edges themselves. Outputs are routed according to the desired acyclic orientation. Each Edge Controller will reset the edge on the node that ended its operation and, at the same time, will set the edge orientation towards the corresponding neighboring node. Figure 8 shows the one possible schematics for the Node and Edge Controllers.

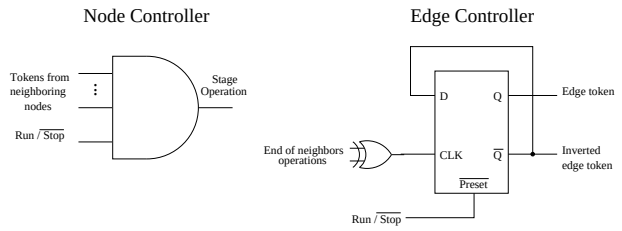

Fig. 8. Schematic for the Node and Edge Controllers.

As demonstrated in [2], it is always possible to draw a SER dynamics mimicking a particular SMER dynamics by representing each node of the SMER multigraph by multiple nodes in the target SER system (once a proper initial acyclic orientation is set). Based on this transformation, SMER-based artificial *central pattern generators* have been implemented in FPGAs [9]. The Node and Edge Controllers shown above were used to convert the multigraphs shown in Figure 4 and Figure 6 into digital circuits. The waveforms shown on Figure 9 represent the output from a CircuitMaker Software TTL implementation of (a) two bits Non-decreasing 2^2 Module counter and (b) of a two bits Gray counter, respectively. Figure 9 (c) shows the waveforms of a three bit Ring counter.

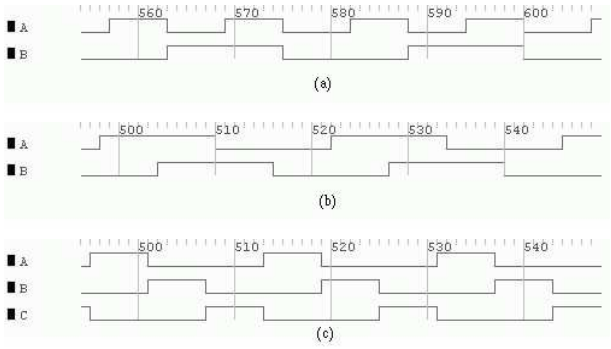

Fig. 9. Waveforms from CircuitMaker software for the counters simulation.

Besides the TTL implementation, an FPGA implementation of the self-timed digital counters are currently being developed. Figure 10 and Figure 11 shows the waveforms and the synthesis properties respectively, of a SMER-based three bits Ring counter implemented in a EPF10K20RC240-4 ALTERA FPGA.

5. CONCLUSION

This work introduced a new way of building self-timed versions of classical counters: Ring counter, Non-decreasing 2^n Module counter and Gray counter. The new methodology is based on both SMER and SER graph dynamics and

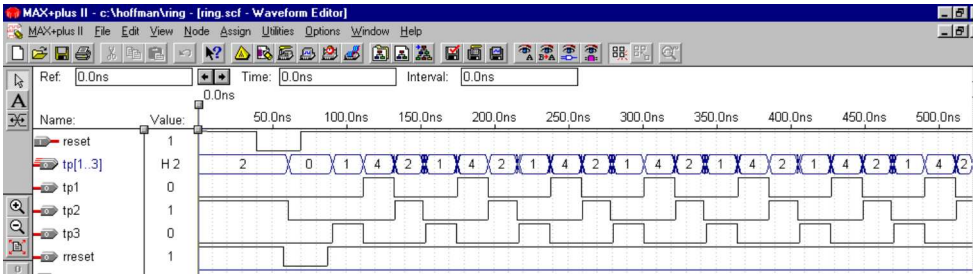

Fig. 10. Ring counter FPGA waveform.

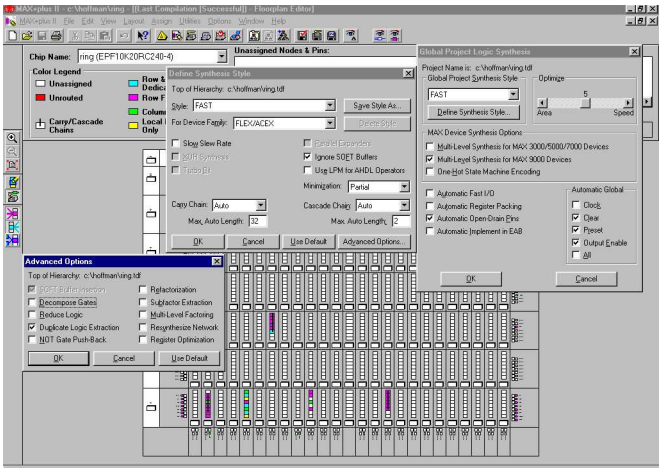

Fig. 11. Ring counter FPGA synthesis.

allows the use of traditional logic gates only. The automatization of SMER-based self-timed sequence generators from pre-specified state diagrams is inside of our interests for future work.

6. REFERENCES

[1] F. M. G França, "Scheduling weightless systems with self-timed boolean networks," *Workshop on Weightless Neural Networks*, pp. 87–92, 1993.

[2] V. C. Barbosa, M. R. F. Benevides, and F. M. G. França, "Sharing resources at nonuniform access rates," *Theory of Computing Systems*, vol. 34, pp. 13–26, 2001.

[3] E. M. Gafni and D. P. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," *IEEE Trans. Commun. COM-29*, vol. 1, pp. 11–18, 1981.

[4] E. W. Dijkstra, "Cooperating sequential processes," *Programming Languages*, pp. 43–112, 1968.

[5] V. C. Barbosa and Gafni E., "Concurrency in heavily loaded neighborhood-constrained system," *ACM Trans. on Prog. Lang. and Systems*, vol. 11, pp. 562–584, 1989.

[6] V. C. Barbosa, *An Introduction to Distributed Algorithms*, MIT Press, 1996.

[7] J.M.S. Alcântara, S.C. Salomão, E.P. Granja, V.C. Alves, and F.M.G. França, "Synchronous to asynchronous conversion - a case study: The blowfish implementation," *Proc. of X IFIP International Conference on VLSI*, pp. 173–180, 1999.

[8] F.M.G. França, V.C. Alves, and E.P. Granja, "Edge reversal-based asynchronous timing synthesis," *Proc. of the IEEE International Symposium on Circuits and Systems*, vol. 89, pp. 45–48, 1998.

[9] Z. Yang and F. M. G. França, "A generalised locomotion *CPGs* architecture based on oscillatory building blocks," *Biological Cybernetics*, vol. 89, pp. 34–42, 2003.