Performance-preserving clustering of elastic controllers

Josep Carmona, Jorge Júlvez, Jordi Cortadella
Universitat Politècnica de Catalunya
Barcelona, Spain

Michael Kishinevsky
Intel Corporation
Hillsboro, USA

ABSTRACT

Asynchronous and latency-insensitive circuits offer a similar form of elasticity that tolerates variations in the delays or the latencies of the computation and communication resources of a system. This flexibility comes at the expense of including a control layer that synchronizes the flow of information. This paper proposes a method for reducing the complexity of the control layer by clustering controllers with similar functionality. The approach reduces the control layer and the number of elastic buffers to a significantly smaller elastic skeleton that preserves the performance of the system. The method also takes into account layout information, thus avoiding optimizations that can be physically unfeasible. The experimental results indicate that drastic reductions in the complexity of the control can be obtained.

1. INTRODUCTION

Asynchronous systems operate without global clock and rely on local handshakes between components for coordination of the computation. They are typically constructed for correct operation under delay changes in their components. Therefore asynchronous systems are often called "elastic" (e.g., "elastic pipelines" [18]) in reference to their flexibility in changing component delays.

Latency insensitive (or *synchronous* elastic) systems have been suggested by a few research groups as a form of discretized asynchronous systems (see, e.g., [3, 6, 8]). Such systems are elastic in the sense that they can tolerate dynamic and static changes in *latencies* of computation and communication components as counted in the number of clock cycles. Their implementation relies on synchronous handshakes optimized for use in coordination with the clock.

The behavior of elastic systems (regardless of synchronous or asynchronous implementation details) and their controllers can be formally described using different concurrent models. Most often Petri Nets and their subclasses are used for this purpose. In particular, if a system has no choice in selecting different branches of behavior (a typical case in elastic systems) its behavior can be described using a Marked Graph (MG).

Fig. 1(a) shows an example of an MG describing the behavior of an elastic system. The transitions of the graph (drawn as boxes) represent events in the system (e.g., computation within the blocks of the data-path ending in latching the results into sequential elements). For convenience, transitions are labeled with names $a, b, \ldots, h$. Every pair of two adjacent transitions, e.g., a and b, is connected with two complementary arcs going in the opposite direction: a forward arc, ab, and a backward arc, ba. For ease of distinguishing within the paper, we draw forward arcs with solid lines, while backward arcs with dotted lines. The pair of complementary arcs is needed to capture handshakes in pipelined behavior: the forward arcs represent the flow of valid information in the system (requests or valid bits), while the backward arcs represent the flow of back-pressure (acknowledgement by the receiver or stop bits). The arcs of an MG are marked with tokens (drawn as black dots) that represent the state of the system. Every transition that has tokens on all input arcs is enabled and can fire by removing one token per input arc and adding one token per output arc. Different enabled transitions can fire concurrently. Each transition has associated the time delay it requires for firing. In asynchronous systems the delay is a real number (or a $[min, max]$ interval of delays). In synchronous systems the delay is a natural number representing the latency of the computation (one unit if operations are single cycle).

Semantically, a pair of complementary arcs between two adjacent transitions, e.g., ab and ba, represent the state of an elastic FIFO placed in the implementation between the two events a and b. The total number of tokens on this pair of arcs is an invariant that corresponds to the capacity of the FIFO. In this paper the capacity is assumed to be always equal to two that corresponds to a case of a 2-slot FIFO (such a FIFO can be implemented using a standard master-slave structure [6]). However cases with different capacity can be treated exactly in the same way. The two tokens on the complementary arcs can be distributed in three possible ways: $(0,2)$, $(1,1)$, and $(2,0)$. The $(0,2)$ (see f,g pair) corresponds to an empty FIFO - we say that there is a *bubble* in the system; the $(1,1)$ (see a,b pair) corresponds to a half-full FIFO that has one token of information, an *info-token* for short; and the $(2,0)$ (see g,h pair) corresponds to a full FIFO that keeps two info-tokens.

The implementation of an elastic system maps a Marked Graph (like the one in Fig. 1) to an asynchronous or a synchronous control circuit. The complexity of the circuit (as measured for example in number of gates) is typically linear in the size of the MG (e.g., [6]). Therefore reducing the size of an MG contributes directly to the size reduction of the control circuit. Based on this fact, we focus on reducing the number of arcs in an MG modeling an elastic system as this reduces the number of elastic FIFOs and the number of channels in the fork and join controllers (that corresponds to transitions with multiple fan-out and fan-in, respectively). For example, Fig. 2 corresponds to sharing of transitions b and f into

e-mail addresses: jcarmona@lsi.upc.edu, julvez@lsi.upc.edu, jordicf@lsi.upc.edu, michael.kishinevsky@intel.com
.

a single transition. As a result of this sharing the implementation is simplified by removing one controller, one channel (a pair of handshake wires), and one elastic FIFO, $F3$, in the data-path that is shared with $F2$.

Many reductions of the modeling MG lead to a functionally equivalent system, but not all of them preserve the performance of the system. The goal of this paper is to identify a class of transformations that reduce the number of controllers while preserving the performance of the system.

The performance of an MG can be measured by its throughput that is defined as the minimal ratio of the number of tokens to the delay across all simple cycles and can be efficiently computed [7]. Assuming that the delays of all transitions in Fig. 1 are equal to 1, the critical cycle is $\{a,b,c,d,e\}$ with a throughput $2/5$ (2 tokens on the arcs of the cycle; delay of the cycle is 5 units). Since the initial marking of arc pairs between a and b and a and f are the same, it is possible to merge transitions b and f (as shown in Fig. 1(b)) without affecting correctness of computation. The throughput of the system is the same $2/5$ and so is the critical cycle $\{a,\{b,f\},c,d,e\}$. In most practical cases, it makes sense to merge b and f only if they are close enough in the layout. This requirement will be fulfilled by clustering only the sets of transitions that are closer than a given physical distance.

Focusing on the new fork transition $\{b,f\}$ we again determine that the initial marking of arc pairs between $\{b,f\}$ and its successors c and g is the same and therefore it is possible to merge transitions c and g (as shown in Fig. 1(c)). However, the throughput of the system reduces to $1/3$ with a new critical cycle $\{\{c,g\},d,h\}$.

Section 2 introduces some basics of MG theory. Section 3 presents sufficient conditions for merging pairs of transitions without reducing the performance of a system. Based on such conditions an efficient strategy is derived (Section 4) to obtain the arcs that can share a controller. Section 4.2 shows how the physical features of the circuits is handled in order to share only those controllers that are closer than a given distance. Section 5 presents results of experiments.

Related work

A few authors demonstrated controller optimization techniques for synchronous elastic systems with static latencies. In [1,4] the technique relies on replacing the handshake controllers with an iteration scheduler that decides when to fire transitions. In [16] it relies on properly aligning all computation branches by a bubble insertion such that back-pressure wires and controllers becomes redundant.

These techniques are complementary to ours and can be jointly

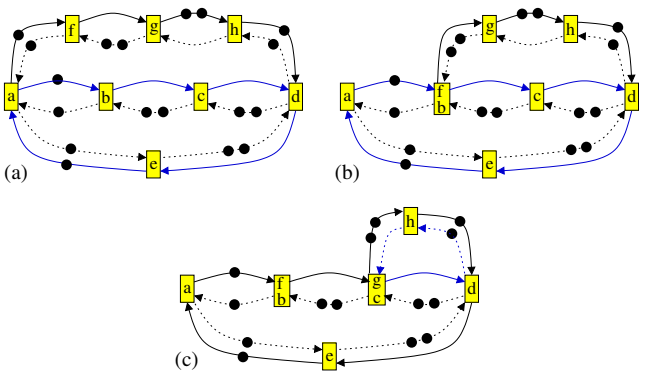

Figure 1: An example of a marked graph (a), merging b and f (b), and c and g (c).

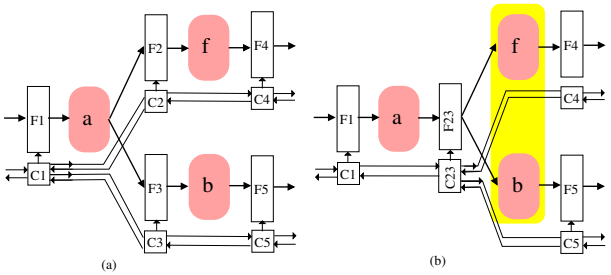

Figure 2: Sharing a controller and an elastic FIFO.

applied. In the context of such techniques our algorithm can be viewed as simplifying the scheduler.

2. MARKED GRAPH MODEL

This section presents a model of timed marked graphs that is used for modeling elastic systems. Although the paper is self-contained the reader can be referred to [13] for a survey on Petri Nets.

2.1 Timed Marked Graphs

DEFINITION 1. *A* Timed Marked Graph *(TMG) is a tuple* $G=(T,A,M_0,\delta)$, *where* T *is a set of transitions (also called nodes),* A *is a set of directed arcs,* $M_0 : A \to \mathbb{N}$ *is a marking that assigns an initial number of tokens to each arc, and* $\delta : T \to \mathbb{R}$ *assigns a non-negative delay to every transition.*

Given a transition $t \in T$, ${}^\bullet t$ and $t^\bullet$ denote the set of incoming and outgoing arcs of t, respectively. Given an arc $a \in A$, ${}^\bullet a$ and $a^\bullet$ refer to the source and target transition of a respectively. A transition t is *enabled* at a marking M if $M(a) > 0$ for every $a \in {}^\bullet t$. An enabled transition t fires after $\delta(t)$ time units. The firing of t removes one token from each input arc of t, and adds one token to each output arc of t.

For the sake of readability it will be assumed in this paper that every transition has unit delay, i.e., $\delta(t)=1$. When dealing with non-unit delays, the obtained equations and conditions are similar but scaled by the values of $\delta(t)$. Subsection 3.3 describes the extension for non-unit delays.

Given a subset $\phi \subseteq A$, the total number of tokens of the arcs in ϕ at a given marking M is denoted by

$$M(\phi) = \Sigma_{a\in\phi} M(a)$$

Given a subset $\mu \subseteq T$ the total delay of μ is denoted by

$$\Delta(\mu) = \Sigma_{t\in\mu} \delta(t)$$

Given a cycle c the total number of tokens and the delay of a cycle are defined as: $M(c) = \Sigma_{a\in c} M(a)$ and $\Delta(c) = \Sigma_{t\in c}\delta(t)$.

2.2 Place invariants and state equation

A few useful properties of strongly connected MGs are as follows [13]:

Token preservation and reachability. We say that the marking M is *reachable* from M_0 if there is a sequence of transitions that can fire starting from M_0 leading to M. Let c be a cycle of a strongly connected MG. A marking M is reachable iff $M(c) = M_0(c)$ for every cycle c.

In the MG shown in Fig. 1(a), there are 15 simple cycles: 9 corresponds to pairs of complementary arcs (each preserving two

tokens), while 6 are longer cycles. Two examples of longer cycles are the critical cycle $\{a,b,c,d,e\}$ that always keeps two tokens and its complement $\{a,e,d,c,b\}$ that always keeps eight tokens.

State equation. Let $\mathbb{C}$ be the $n \times m$ incidence matrix of the MG with rows corresponding to n arcs and columns to m transitions.

$$\mathbb{C}_{ij} = \begin{cases} -1 & \text{if } t_j \in a_i^\bullet \setminus {}^\bullet a_i \\ +1 & \text{if } t_j \in {}^\bullet a_i \setminus a_i^\bullet \\ 0 & \text{otherwise} \end{cases}$$

If marking M is reachable from the initial marking M_0 of an MG then the state equation

$$M = M_0 + \mathbb{C} \cdot \sigma \tag{1}$$

is satisfied for some firing count vector σ (the j's component of σ corresponds to the number of times transition t_j has fired).

2.3 Elastic Marked Graphs

DEFINITION 2. *An* Elastic Marked Graph *(EMG) is a TMG such that for any arc $a \in A$ there exists a complementary arc $a' \in A$ satisfying the following condition ${}^\bullet a = a'^\bullet$ and ${}^\bullet a' = a^\bullet$.*

A labelling function L maps all arcs of an EMG as forward or backwards $L : P \to \{F,B\}$ such that $L(a) = F$ iff $L(a') = B$.

Due to the existence of complementary arcs, an EMG is strongly connected iff it is connected. Since a non-connected EMG can be seen as a set of separated strongly connected EMGs, all the EMGs considered here are assumed to be strongly connected. Each pair of complementary arcs a and a' correspond to a cycle with two arcs and therefore the number of tokens on them is preserved. As explained in the introduction, in this paper we assume that for any complementary a and a' $M(\{a,a'\}) = 2$. Therefore, all EMGs in this paper are 2-bounded (an arc cannot have more than two tokens).

Without loss of generality, we model elastic systems with strongly connected EMGs. For open systems interacting with an environment, it is possible to incorporate an abstraction of the environment into the model by a transition that connects the outputs with the inputs.

2.4 Throughput of an EMG

The performance of an EMG can be measured by the throughput of its transitions. Given that we are considering strongly connected EMGs, in the steady state all transitions have exactly the same throughput, Θ. The throughput intuitively corresponds to the number of times each operation is performed on average per unit of time during the infinitely long execution of the system.

If C is the set of simple directed cycles in an EMG, its throughput can be determined as:

$$\Theta = min_{c \in C} \frac{M_0(c)}{\Delta(c)} \tag{2}$$

where $M_0(c)$ and $\Delta(c)$ are the number of tokens and the delay of cycle c, respectively [14, 15].

DEFINITION 3 (CRITICAL CYCLE AND ARC). *A cycle* $\mathbf{c}$ *satisfying the equality* $\Theta = \frac{M_0(\mathbf{c})}{\Delta(\mathbf{c})}$ *is called* critical. *An arc is called critical if it belongs to a critical cycle.*

Many efficient algorithms for computing the throughput of an EMG exist that do not require an exhaustive enumeration of all cycles [7, 9].

2.5 Average marking

The average marking of an arc a, denoted as $\overline{M}(a)$, represents the average occupancy of the arc during the steady state execution.

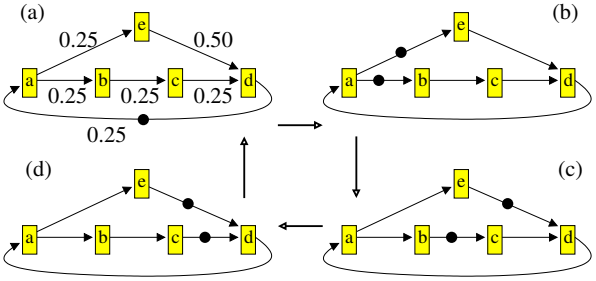

Figure 3: A marked graph and its time evolution.

The system in Fig. 3 (a) has two cycles: $\{a,b,c,d\}$ and $\{a,e,d\}$. Both have one token. When a fires the token is removed from da and two tokens are produced: one at arc ae, and the other at ab (see Fig. 3 (b)). From this new marking transitions b and e are enabled and fire in one time unit yielding the marking in Fig. 3 (c). Now c is enabled, its firing results in the marking in Fig. 3 (d). At this point d is enabled. After it fires the initial marking is reached again. Since the system remains one time unit at each step, it can be deduced that arcs ab, bc, cd, da and ae keep a token during one fourth of the overall time, thus, their average marking is $\overline{M}(ab) = \overline{M}(bc) = \overline{M}(cd) = \overline{M}(da) = \overline{M}(ae) = 0.25$. The arc ed however contains a token during two of the four mentioned steps, therefore $\overline{M}(ed) = 0.5$.

Formally the average marking vector for all arcs is defined as:

$$\overline{M} = lim_{\tau \to \infty} \frac{1}{\tau} \int_0^\tau M(t)dt$$

where τ is the time variable.

Each pair $\{a, a^\bullet\}$ of the EMG can be seen as a simple queuing system for which Little's formula [10] can be directly applied and therefore the following formula holds:

$$\overline{M}(a) = \Theta \cdot \overline{R}(a)$$

where $\overline{R}(a)$ is the average residence time at arc a, i.e., the average time spent by a token on the arc a [2]. The average residence time is the sum of the average waiting time due to a possible synchronization (for instance in Fig. 3, the token in ed has to wait one time unit for the token in cd) and the average service time which in the case of EMGs is $\delta(a^\bullet)$. Therefore the service time $\delta(a^\bullet)$ is a lower bound for the average residence time. This leads to the inequality:

$$\overline{M}(a) \geq \Theta \cdot \delta(a^\bullet) \qquad \text{for every arc } a \tag{3}$$

Due to the above token preservation property of EMG cycles it can be proven that the token preservation property holds not only for any reachable marking, but also for an average marking. That is for any cycle c the sum of tokens in the initial marking $M_0(c)$ and in the average marking $\overline{M}(c)$ is the same. E.g., for the two cycles in Fig. 3: $M_0(a,b,c,d) = \overline{M}(a,b,c,d) = 1$ and $M_0(a,e,d) = \overline{M}(a,e,d) = 1$. In particular, this statement holds for any critical cycle, $\mathbf{c}$: $M_0(\mathbf{c}) = \overline{\mathbf{M}}(\mathbf{c})$. Then, equation (2) can be rewritten for a critical cycle as follows:

$$\Theta = \frac{\overline{M}(\mathbf{c})}{\Delta(\mathbf{c})} \tag{4}$$

Combining expressions (3) and (4) yields the following equation for every critical arc **a**:

$$\overline{M}(\mathbf{a}) = \Theta \cdot \delta(\mathbf{a}^\bullet) \tag{5}$$

For unit delay transitions, equation (5) states that the average marking of every critical arc is equal to the throughput.

Similarly, the state equation (1) can be expanded to real domain for markings M and firing vectors σ and is, in particular, satisfied by the average marking $\overline{M}$.

$$\overline{M} = M_0 + \mathbb{C} \cdot \sigma, \text{where } M \in \mathbb{R}^{|A|} \text{ and } \sigma \in \mathbb{R}^{|T|} \tag{6}$$

3. EMG REDUCTION

Let us consider the system depicted in Fig. 3(a) and its time evolution (b), (c) and (d). The arcs *ab* and *ae* have the same input transition *a*, thus, a token will always appear at both arcs simultaneously. Moreover, since their output transitions have the same delay *b* and *e* always fire simultaneously. Hence, arcs *ab* and *ae* always have the same marking, and as a consequence also have the same average marking, $\overline{M}(ab) = \overline{M}(ae)$. Therefore transitions *b* and *e* can be merged into a single transition without decreasing the throughput of the system. In the elastic implementation this implies that one controller suffices to control both *ab* and *ae*.

Based on the above intuition let us present the sufficient conditions for merging transitions and arcs of EMGs preserving system performance.

3.1 Tight marking

Fork transitions (like *a* in Fig. 3(a)) are potential sources of arcs with same average markings, *ab* and *ae*, since fork transitions serve as synchronization points. This section describes a polynomial algorithm for calculating a special marking (called a *tight* marking) that, if possible, assigns the same marking value to the output arcs of the fork transitions.

DEFINITION 4. *A marking $\tilde{M}$ such that $\tilde{M} \in \mathbb{R}^{|A|}$ is called a* tight *marking of an EMG if it satisfies the inequality* (3) *and the state equation* (6) *and for every transition t there exists an arc $a \in {}^\bullet t$ such that $\tilde{M}(a) = \Theta$, where Θ is the throughput of the EMG. An arc a satisfying condition $\tilde{M}(a) = \Theta$ is called* tight.

Since a tight marking satisfies (3) and (6), each critical arc *a* is necessarily tight, i.e., $\tilde{M}(a) = \Theta$. On the other hand, non critical arcs have some slack to verify (3) and (6). The tight marking exploits this flexibility by adjusting the marking value for some arcs, at least one per transition, to the systen throughput. This tight making eases the formulation of sufficient conditions to merge transitions.

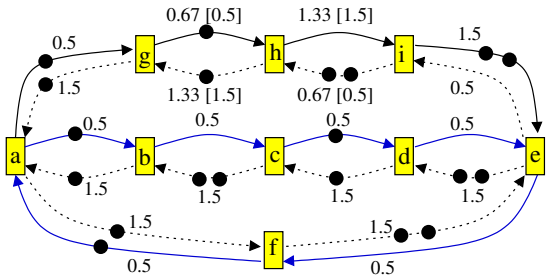

Figure 4: An EMG illustrating a tight marking.

Let us consider the EMG in Fig. 4. It has a single critical cycle $\{a,b,c,d,e,f\}$ with a throughput 0.5. Each arc in Fig. 4 is labeled with one number if its average and tight markings coincide. In cases when they are different the average marking is listed first and the tight marking is shown in square brackets. It can be seen that the tight marking satisfies all the conditions of Definition 4.

LEMMA 1. *For any EMG there exists a tight marking $\tilde{M}$.*

Proof: See Appendix.

PROPOSITION 2. *A tight marking of a unit-delay EMG can be computed by solving the following Linear Programming (LP) problem that includes the constraints from* (3) *and* (6) *:*

$$\begin{aligned} &\textit{Maximize } \Sigma\sigma : \\ &\quad \Theta \le \tilde{M}(a) \quad \textit{for every } a \in A \\ &\quad \tilde{M} = M_0 + \mathbb{C} \cdot \sigma \\ &\quad \sigma(t) \le k \quad \textit{for every } t \in T \end{aligned} \tag{7}$$

where $k \in \mathbb{R}$ is any real number.

Proof: See Appendix.

The last constraint guarantees that the firing sequence σ is bounded. The first two constraints of (7) can be transformed into a single one:

$$\Theta \cdot \mathbf{1} - M_0 \le \mathbb{C} \cdot \sigma \tag{8}$$

Since we are dealing with MGs, each row of the incidence matrix $\mathbb{C}$ contains a single positive (1) and a single negative (-1) value, while all other values are zeros. Therefore, equation (8) is a system of *difference constraints* and hence the LP (7) can be efficiently solved by the Bellman-Ford algorithm with the time complexity of $O(|T||A|)$ [5].

3.2 Merging transitions

The basic transformation in reduction of an EMG is merging a pair of transitions. Merging two transitions t_i and t_j in an EMG $G = (T, A, M_0, \delta)$ leads to a new EMG $G < t_i, t_j >= (T', A', M_0', \delta')$ in which two transitions t_i and t_j are replaced with a new one t_{ij}. All input and output arcs of t_i and t_j are replaced with input and output arcs of t_{ij} such that $a \in {}^\bullet t_{ij}$ iff $(a \in {}^\bullet t_i \vee a \in {}^\bullet t_j)$ and $a \in t_{ij}^\bullet$ iff $(a \in t_i^\bullet \vee a \in t_j^\bullet)$. The initial marking for a new arc is equal to the initial marking of the corresponding replaced arc. The delay of the merged transition is the maximum of delays of the merged transitions, $\delta(t_{ij}) = max\{\delta(t_i), \delta(t_j)\}$. After merging two transitions a multi-graph is obtained since two transitions can be connected by more than one arc. If the new EMG $G < t_i, t_j >$ has two identical arcs v and w, i.e., $M_0'(v) = M_0'(w)$, $L(v) = L(w)$, ${}^\bullet v = {}^\bullet w$ and $v^\bullet = w^\bullet$, then v and w can be merged into a single arc.

Fig. 1 gives an example of reducing an original EMG by merging transitions *b* and *f*. After merging this pair of transitions, identical arcs have also been merged into one arc, see Fig. 1(b).

For convenience our algorithm reduces an EMG in two steps: first some transitions are merged, then identical arcs are merged.

DEFINITION 5. *Transitions t_i and t_j are called* mergeable *if an EMG $G < t_i, t_j >$ obtained by merging transitions t_i and t_j in an EMG G has the same throughput as G.*

The following theorem forms a basis for the algorithm of reducing EMGs.

THEOREM 3. *Transitions t_i and t_j in an EMG are mergeable if there exist arcs $a_i \in {}^\bullet t_i$ that contains only the tight arcs of the system.and $a_j \in {}^\bullet t_j$ such that:*

- $L(a_i) = L(a_j)$,
- $\tilde{M}(a_i) = \tilde{M}(a_j) = \Theta$,
- *(${}^\bullet a_i = {}^\bullet a_j$) or (${}^\bullet a_i$ and ${}^\bullet a_j$ are mergeable).*

Proof: See Appendix.

The first two conditions of Theorem 3 narrow the search space to tight arcs with the same label (forward or backward). The third condition condition defines iterative merging. These three conditions ensure the existence of an initialization, i.e., firing sequence of transitions, that produces a marking M in which $M(a_i) = M(a_j)$ (see the proof of the Theorem 3 in the Appendix). After such initialization, transitions t_i and t_j can effectively be merged. This merging will make arcs a_i and a_j be identical, since $M(a_i) = M(a_j)$, $L(a_i) = L(a_j)$, ${}^\bullet a_i = {}^\bullet a_j$ and $a_i^\bullet = a_j^\bullet$, and hence they will be merged into a single arc.

Fig. 5 shows the tight arcs of the EMG in Fig. 4. The only remaining cycle is the critical one. Transitions b and g in Fig. 4 do fulfill the conditions of Theorem 3 and therefore they are mergeable. Moreover, since $M_0(ab) = M_0(ag)$ transitions b and g can be merged without any initialization. Transitions c and h also fulfill the conditions. Given that $M_0(bc) \neq M_0(gh)$, an initialization is required before merging c and h. In this case, firing h is enough to initialize the system: such firing removes one token from gh and ih, and adds one token in hg and hi. This way, a marking M is obtained in which $M(bc) = M(gh)$. Now transitions c and h can effectively be merged.

The system in Fig. 4 shows why the tight marking captures better the flexibility for merging transitions than the average marking. The arc gh is tight since $\tilde{M}(gh) = 0.5$. However it is not critical since $\overline{M}(gh) = 0.67$. Therefore, transitions c and h could not get merged if the average marking is used in place of the tight marking in Theorem 3.

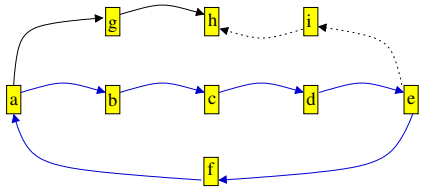

Figure 5: Tight subgraph of the system from Fig. 4.

One can also consider the scenario in which the initial marking is fixed, that is, no initialization is allowed. In such case, in addition to the conditions in Theorem 3, the condition $M_0(a_i) = M_0(a_j)$ must hold to merge transitions t_i and t_j. As might be expected, this new constraint reduces the number of transitions that can be merged with respect to the previous scenario (see Section 5).

3.3 Non-unit delays

The reduction method based on merging transitions and arcs described in the previous subsection apply to EMGs with unit delay transitions, $\delta(t) = 1$. It is easy to extend these results for arbitrary delays: the value of $\delta(t)$ becomes a scaling factor of the throughput (as derived from (3)). In particular, the last condition of Definition 4 must be changed to $\tilde{M}(a) = \delta(t) \cdot \Theta$, the constraint $\Theta \leq \tilde{M}(a)$ of LP (7) must be changed to $\delta({}^\bullet a) \cdot \Theta \leq \tilde{M}(a)$ and inequality (8) should change to $\delta \cdot \Theta - M_0 \leq \mathbb{C} \cdot \sigma$.

An easy way to make Theorem 3 still applicable is by adding the new condition $\delta(t_i) = \delta(t_j)$. This condition however does not always hold. Nonetheless, it is still possible to merge two transitions t_i, t_j with $\delta(t_i) > \delta(t_j)$ simply by adding an extra delay $\delta(t_i) - \delta(t_j)$ to t_j. Notice that in some cases it is not possible to increase a transition delay without violating inequality (3), i.e., without decreasing the throughput. Thus, if the throughput must remain the same, one have to search for the pairs of transitions that satisfy (3) even after the delay increase.

4. HEURISTICS FOR REDUCING AN EMG

The overall strategy for reducing an EMG involves the following steps:

1. Computation of the throughput of the system.
2. Computation of a tight marking (Subsection 3.1).
3. Determine sets of mergeable transitions (Theorem 3) by traversing the tight subgraph and merge them. A good heuristics is selecting critical fork transitions and exploring tight arcs at the output of these transitions. Every set of mergeable transitions then becomes a new starting point for the search of the next set and the method iterates until the fixed point.
4. Fire transitions to obtain the same marking in the input arcs of the mergeable transitions.
5. Merge mergeable transitions and identical arcs.

We further illustrate the above steps by an example.

4.1 Example

Fig. 6 depicts the associated marked graph of the largest strongly connected component obtained for a logic circuit s27 of the MCNC benchmark suite. For the sake of clarity only forward arcs ($L(a) = F$) are shown. There is one backward arc a' per every forward arc a such that $M_0(a) + M_0(a') = 2$ that are not drawn in the figure. The initial marking is randomly defined.

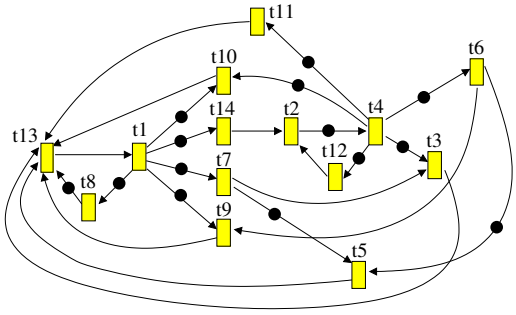

Figure 6: Forward arcs of an EMG corresponding to circuit s27.

Let us assume the flexible marking scenario.

Throughput computation. The throughput of the system is 0.25. There exists only one critical cycle: $\{t_1, t_7, t_3, t_{13}\}$.

Tight marking computation. Once a tight marking $\tilde{M}$ is computed, the proposed strategy visits tight arcs (i.e. arcs a with tight marking equal to the throughput $\tilde{M}(a) = \Theta$). The tight subgraph is

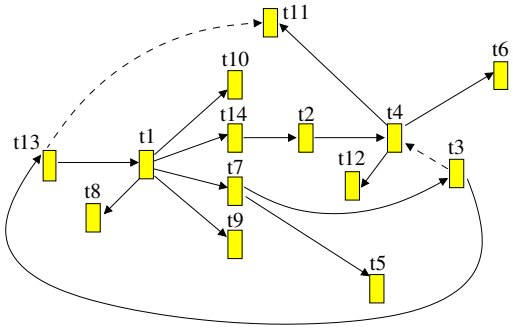

Figure 7: Tight arcs of s27.

iteration	trans. merged	trans. fired
1	$t_7, t_8, t_9, t_{10}, t_{14}$	none
2	t_2, t_3, t_5	t_5
3	t_4, t_{13}	t_4
4	t_1, t_6, t_{11}, t_{12}	$2 \cdot t_6, 2 \cdot t_{11}, 2 \cdot t_{12}$

Table 1: Transitions merged in each iteration and fired for changing the initial marking.

shown in Fig. 7. Notice that the tight arcs can include both forward and backward arcs, e.g., arcs t_3t_4 and $t_{13}t_{11}$ are backward.

Merging transitions. Each transition of the critical cycle is a fork in the subgraph of Fig. 7, thus, any of them can be selected to start the search. Let us pick-up t_1. Transition t_1 has five outgoing arcs t_1t_7, t_1t_8, t_1t_9, t_1t_{10} and t_1t_{14} all with the tight marking equal to the throughput and all labelled with the same label. These transitions fulfill the conditions of Theorem 3. Given that all those arcs have the same initial marking, transitions t_7, t_8, t_9, t_{10} and t_{14} can be merged into a single transition without decreasing the system throughput. Next, the algorithm takes the tight successor arcs of the merged transitions: $t_{14}t_2$, t_7t_3 and t_7t_5. However, their initial markings are not the same: $M_0(t_{14}t_2) = M_0(t_7t_3) = 0$, while $M_0(t_7t_5) = 1$. Transition t_5 can fire yielding a modified initial marking such that $M_0(t_{14}t_2) = M_0(t_7t_3) = M_0(t_7t_5)$. Now, transitions t_2, t_3 and t_5 can be merged.

During the next iteration, the tight arcs output to merged transitions are: t_2t_4, t_3t_{13} and t_3t_4. One of these arcs is backward (t_3t_4), hence, there is no possibility of merging its output transition t_4 with other transitions. By firing t_4 the initial markings of t_2t_4, t_3t_{13} are made equal and therefore transitions t_4 and t_{13} can be merged. Finally, the output tight arcs of t_4 and t_{13} are t_4t_6, t_4t_{11}, t_4t_{12}, $t_{13}t_1$, $t_{13}t_{11}$. The backward arc $t_{13}t_{11}$ is discarded. In order to have the same marking on the remaining forwarded arcs transitions t_6, t_{11} and t_{12} are fired twice each. Now, transitions t_1, t_6, t_{11} and t_{12} can be merged. Since all transitions have been visited no further transitions should be examined and the algorithm converges.

Table 1 summarizes the sets of transitions merged by the algorithm during each iteration. The algorithm has merged all transitions in four new transitions: $a = \{t_7,t_8,t_9,t_{10},t_{14}\}$, $b = \{t_2,t_3,t_5\}$, $c = \{t_4,t_{13}\}$ and $d = \{t_1,t_6,t_{11},t_{12}\}$. As a result every original transition got mapped to a transition on the critical cycle. It can be checked that the result is the same if the first visited transition is other than t_1.

Merging arcs. The sets of forward arcs that can be merged after merging transitions are as follows: $A_a = \{t_1t_{10},t_1t_{14},t_1t_8,t_1t_9,t_1t_7\}$, $A_b = \{t_4t_{12},t_4t_{11},t_4t_6,t_{13}t_1\}$, $A_c = \{t_{14}t_2,t_7t_3,t_7t_5\}$, $A_d = \{t_2t_4,t_3t_{13}\}$, $A_e = \{t_{10}t_{13},t_9t_{13}\}$, $A_f = \{t_{12}t_2,t_6t_5\}$, $A_g = \{t_8t_{13}\}$, $A_h = \{t_4t_{10}\}$, $A_i = \{t_4t_3\}$, $A_j = \{t_{11}t_{13}\}$, $A_k = \{t_6t_9\}$, and $A_l = \{t_5t_{13}\}$. The sets of complementary arcs, $A'_a,\ldots,A'_l$, corresponding to the backward arcs and their controllers can also be merged. The net resulting of merging the mergeable transitions and merging identical arcs is shown in Fig. 8. In such figure only the forward arcs are depicted, the marking is the one obtained after initialization. Notice that in Fig. 8 there are some redundant arcs, e.g., T_dT_a arc with two tokens, that can be removed without decreasing the throughput. Nevertheless, not all redundant arcs have the same initial marking as the arcs with the same input and output transition. For this reason they explicitly need a different controller.

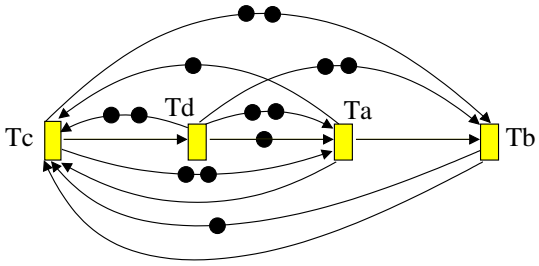

Figure 8: Net after merging transitions and arcs.

4.2 Layout-aware optimization

The clustering method previously presented merges transitions and arcs ignoring layout information about the circuit implementation of the corresponding elastic circuit. It can result in merging elastic controllers that are separated by large physical distances in the layout, thus creating long handshake wires that can have a negative impact on the performance of the system. It is desirable that the optimization of controllers is performed according to the physical placement of the circuit components.

Incorporating layout information in the clustering approach can be done in two different ways.

1. The layout information can be used to restrict the merging of controllers when they are far apart, or
2. The layout unaware clustering can first be performed followed by a split of large clusters into smaller ones according to the proximity of the controllers.

By experimentation we found the latter approach more effective since it has a lower dependency on the merging order of the controllers. The method used for splitting each cluster into a set of sub-clusters is based on the well-known k-means clustering algorithm [11].

Given the longest allowed distance d, the k-means clustering algorithm produces clusters whose *smallest enclosing circle* has radius d. Checking this condition can be done in linear time [12].

The algorithm for splitting clusters is as follows:

```
If radius of cluster is <= d then  exit.
c := 2;
repeat forever
  Split cluster into c sub-clusters
       using the k-means algorithm;
  If radius of each sub-cluster
       is  <= d then exit.
  c := c+1;
end repeat
```

5. EXPERIMENTAL RESULTS

Elasticity and clustering are concepts that must be applied at a coarse level of granularity in which, for example, each computational block can have several hundreds or thousands of gates and each register can be 32- or 64-bit wide. Under the absence of a representative set of benchmarks with elastic controllers and physical information, we decided to create an experimental setup based on existing graphs in the literature.

5.1 Experimental setup

The underlying graphs of the sequential ISCAS benchmarks were interpreted as coarse-level netlists. Each gate was interpreted as a computational block with unit delay (transition in the EMG), whereas each edge was interpreted as a communication channel (arc in the EMG). Moreover, each channel was supposed to have a half-full FIFO with capacity 2.

To generate physical information for an n-block netlist, a unit square grid with $s \times s$ cells, $s = \lceil \sqrt{n} \rceil$, was generated. All the blocks were assumed to have unit size and were placed on the layout using Capo [17].

After placement, wire pipelining was applied to those channels longer than d units (d is a parameter of the experiments). Empty FIFOs with capacity 2 were inserted in such a way that the length of each channel did not exceed d (see Fig. 9). It is important to realize that the insertion of empty FIFOs reduces the throughput of the system when the channel is in one of the critical loops of the system.

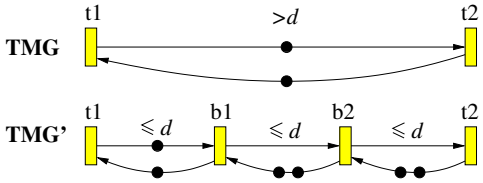

Figure 9: Wire pipelining to reduce the length of long channels.

5.2 Results

The elastic netlists obtained with the previous approach were optimized using the algorithm presented in Section 4. The results are reported in Table 2. Different optimization scenarios were considered:

(a) **Opt.** ($d = \infty$): no constraints are imposed on the size of each cluster. The initial state is not allowed to change.

(b) **Opt. & Init** ($d = \infty$): no constraints are imposed on the size of each cluster. The initial state is allowed to be moved (retimed) in case it is convenient for optimization.

(c) **Placement & Opt. & Init** ($d < \infty$): each cluster is constrained to be enclosed in a circle of radius d. The initial state is allowed to change. The algorithm presented in Section 4.2 is used for cluster splitting. The value $d = 10$ was used for all benchmarks, except for the last two (s38417 and s38584) in which the value $d = 20$ was used to avoid an excessive degradation of the throughput.

The columns nodes/arcs report the size of the examples before and after optimization. The column Θ reports the throughput of each example after having inserted the empty FIFOs for wire pipelining. The column `n.size` indicates the normalized size of the graph (nodes+arcs) after optimization with regard to the original graph (size=1). The runtime is only reported for the last scenario, which is the most CPU consuming. It is important to emphasize that most of the runtime is spent on the execution of the k-means clustering algorithm for partitioning large clusters.

5.3 Discussion

In general, the results indicate that the control layer of elastic systems, either synchronous or asynchronous, can be highly optimized by loosing some elasticity but still maintaining the same performance. By looking at the first scenario (Opt. with $d = \infty$), we can observe a wide range of optimizations, from a drastic reduction (`s298`) to more moderate reductions (about half size for `s9234`).

The reductions are even more remarkable when allowing to change the initial state by initially firing some transitions (Opt. & Init.), where in most cases the size of the graph is less than 10% of the size of the original graph. The effectiveness of this approach relies on the possibility of changing the initial state, which is a similar problem of calculating the initial state after retiming [19]. This scenario gives an estimation of the maximum controller reduction that can be achieved when no physical constraints are imposed.

Finally, the last scenario shows the impact of incorporating physical constraints on the clustering of controllers. We observe that the reductions are a little bit smaller, but still important. The major impact of the physical constraints are observed on the largest examples.

We have to bear in mind, that our graphs intend to represent systems with large computational blocks. In real-life examples, we should expect graphs with dozens or few hundreds of blocks. Therefore, the type of optimizations that we can expect would correspond to the smallest examples in the table.

6. CONCLUSIONS

This paper has shown that elasticity can be incorporated in conventional circuits at the expense of a small control cost. While still having distributed controllers along the system, the network can be significantly reduced by clustering controllers.

Even though the approach presented in the paper has been evaluated for synchronous circuits, it can be easily extended and applied to asynchronous circuits, thus bringing similar advantages.

7. REFERENCES

[1] J. Boucaron, J. Millo, and R. de Simone. Latency-insensitive design and central repetitive scheduling. In *IEEE-ACM International Conference MEMOCODE'06*, pages 175–183, 2006.

[2] J. Campos and M. Silva. Structural Techniques and Performance Bounds of Stochastic Petri Net Models. In G. Rozenberg, editor, *Advances in Petri Nets 1992*, volume 609 of *Lecture Notes in Computer Science*, pages 352–391. Springer, 1992.

[3] L. Carloni, K. McMillan, and A. Sangiovanni-Vincentelli. Theory of latency-insensitive design. *IEEE Transactions on Computer-Aided Design*, 20(9):1059–1076, Sept. 2001.

[4] M. Casu and L. Macchiarulo. A new approach to latency insensitive design. In *Proc. Digital Automation Conference (DAC)*, pages 576–581, June 2004.

[5] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

				Opt. ($d=\infty$)			Opt. & Init. ($d=\infty$)			Place & Opt. & Init. ($d<\infty$)			
example	**nodes**	**arcs**	**Θ**	**nodes**	**arcs**	**n.size**	**nodes**	**arcs**	**n.size**	**nodes**	**arcs**	**n.size**	**CPU**
s208	159	458	0.50	33	120	0.247	12	32	0.071	12	32	0.071	< 1 s.
s298	9640	40592	0.33	21	92	0.002	9	58	0.001	29	230	0.005	34 m.
s344	296	838	0.50	64	236	0.264	6	22	0.024	8	56	0.056	1 s.
s349	274	774	0.50	63	222	0.271	22	102	0.118	32	214	0.234	1 s.
s382	278	876	0.50	32	104	0.117	9	28	0.032	10	40	0.043	2 s.
s386	131	792	0.50	19	52	0.076	11	34	0.048	11	34	0.048	< 1 s.
s400	264	840	0.50	39	134	0.156	35	112	0.133	38	148	0.168	1 s.
s420	321	954	0.50	102	334	0.341	37	114	0.118	43	194	0.185	2 s.
s444	338	1060	0.50	36	132	0.120	6	22	0.020	10	106	0.082	4 s.
s510	446	1990	0.50	26	86	0.045	10	30	0.016	13	54	0.027	13 s.
s526	315	1128	0.50	30	96	0.087	7	20	0.018	10	74	0.058	5 s.
s641	493	1276	0.50	232	588	0.463	88	224	0.176	101	372	0.267	11 s.
s713	545	1452	0.33	255	692	0.474	159	472	0.315	200	722	0.461	12 s.
s820	325	2220	0.50	31	98	0.057	15	60	0.029	21	124	0.056	13 s.
s832	381	2470	0.50	12	42	0.018	6	18	0.008	10	102	0.039	11 s.
s838	743	2198	0.33	172	704	0.297	47	182	0.077	60	444	0.171	28 s.
s953	921	2686	0.33	345	1274	0.448	78	308	0.107	100	666	0.212	35 s.
s1423	1073	3244	0.33	231	964	0.276	26	106	0.003	43	526	0.019	58 s.
s1488	1127	5718	0.50	13	40	0.007	8	24	0.004	14	106	0.017	3 m.
s1494	1239	6220	0.50	9	28	0.004	6	20	0.003	14	132	0.019	1 m.
s5378	4328	12162	0.14	1871	5774	0.463	1110	4214	0.322	1678	6154	0.474	1 m.
s9234	4219	11200	0.14	2094	5904	0.518	1388	4444	0.378	1858	6198	0.522	58 s.
s38417	32106	87242	0.12	4730	16310	0.176	1154	4884	0.050	3139	26394	0.247	15 h.
s38584	30647	93142	0.12	3991	13978	0.145	759	2688	0.027	1751	19644	0.172	16 h.

Table 2: Results for the MCNC benchmarks.

[6] J. Cortadella, M. Kishinevsky, and B. Grundmann. Synthesis of synchronous elastic architectures. In *Proc. ACM/IEEE Design Automation Conference*, pages 657–662, July 2006.

[7] A. Dasdan and R. K. Gupta. Faster maximum and minimum mean cycle algorithms for system performance analysis. *IEEE Transactions on Computer-Aided Design*, 17(10):889–899, 1998.

[8] A. Edman and C. Svensson. Timing closure through a globally synchronous, timing partitioned design methodology. In *DAC*, pages 71–74, 2004.

[9] R. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[10] J. D. C. Little. A proof of the queueing formula $L = \lambda W$. *Operations Research*, 9:383–387, 1961.

[11] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symp. on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

[12] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12(4):759–776, 1983.

[13] T. Murata. Petri Nets: Properties, analysis and applications. *Proceedings of the IEEE*, pages 541–580, Apr. 1989.

[14] C. V. Ramamoorthy and G. S. Ho. Performance evaluation of asynchronous concurrent systems using petri nets. *IEEE Trans. Software Eng.*, 6(5):440–449, 1980.

[15] C. Ramchandani. Analysis of asynchronous concurrent systems by timed Petri nets. Technical Report Project MAC Tech. Rep. 120, Massachusetts Inst. of Tech., Feb. 1974.

[16] T. Raudvere, I. Sander, and A. Jantsch. A synchronization algorithm for local temporal refinements in perfectly synchronous models with nested feedback loops. In *Proceedings of GLSVLSI'07*, March 2007.

[17] J. A. Roy, D. A. Papa, S. N. Adya, H. H. C. an, A. N. Ng, J. F. Lu, and I. L. Markov. Capo: robust and scalable open-source min-cut floorplacer. In *ISPD '05: Proceedings of the 2005 international symposium on Physi cal design*, pages 224–226, New York, NY, USA, 2005. ACM Press.

[18] I. E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989.

[19] H. Touati and R. Brayton. Computing the initial states of retimed circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 12(1):157–162, 1993.

APPENDIX

Proof of Lemma 1:

Proof: Let $\hat{M}$ be a marking that fulfills $\hat{M}(a) \geq \Theta \cdot \delta(a^\bullet)$ for every arc a (see (3)). Clearly at least one $\hat{M}$ exists that is the average marking. In particular it holds $\hat{M}(a) = \Theta \cdot \delta(a^\bullet)$ for every arc a in a critical cycle (see (5)). Let $T' = \{t |$ for every $a \in {}^\bullet t\ \hat{M}(a) > \Theta \cdot \delta(t)\}$ (note that the transition in the critical cycles are not in T'). Let us fire every transition $t \in T'$ in a quantity $g(t,\hat{M}) = min_{\{a \in {}^\bullet t}\{\hat{M}(a) - \Theta \cdot \delta(t)\}$ obtaining $\hat{M}'$. If the procedure of computing T' and $\hat{M}'$, then T'' and $\hat{M}''$ cannot be repeated indefinitely then the result is proved. If it can be repeated indefinitely, there exists a set of transitions W such that for every $t \in W$ the sequence $g(t,\hat{M})$, $g(t,\hat{M}')$, $g(t,\hat{M}'')$ is not tending to 0. Since $W \subset T$, this would mean that there is a repetitive sequence that does not include all transitions. Contradiction. □

Proof of Proposition 2:

Proof: Since EMG s are repetitive systems, a marking M can be reached with any firing sequence $\sigma + j \cdot \mathbf{1}$ where j is any real number. A constraint like $\sigma(t_a) = k$ ensures boundedness of the solution. Since t_a is in a critical cycle, there exists an arc $a \in {}^\bullet t_a$ such that a is also in the same critical cycle. Hence, the solution of the LP will necessarily verify $\hat{M}(a) = \Theta \cdot \delta(a^\bullet)$ (see equation (5)). Given that the objective function $\Sigma\sigma$ is maximized, for every transition t there will exist $a \in {}^\bullet t$ such that $\tilde{M}(a) = \Theta \cdot \delta(t)$. Hence,

the obtained marking $\tilde{M}$ is a tight marking. □

Proof of Theorem 3:

Proof: Let us assume that there exist t_i, t_j that verify the conditions of the theorem. Since ${}^\bullet a_i$, ${}^\bullet a_j$ are joinable (or ${}^\bullet a_i = {}^\bullet a_j$) we can reason on the graph obtained after merging ${}^\bullet a_i$ and ${}^\bullet a_j$ into a single transition t_x. Assume without loss of generality that $M_0(a_i) \geq M_0(a_j)$. Let us fire transition t_i as many times as $M_0(a_i) - M_0(a_j)$ producing marking M (notice that the EMGs consider in this paper are 2-bounded, and hence, it holds $M_0(a_i) - M_0(a_j) \leq 2$). At M it holds $M(a_i) = M(a_j)$. Since a_i and a_j have the same input transition t_x, if t_i and t_j are merged arcs a_i and a_j will have the same input transition and the same output transition. Thus, a_i and a_j will be identical, i.e., they will always have the same marking. In addition, the fact that $\tilde{M}(a_i) = \tilde{M}(a_j) = \Theta \cdot \delta$ ensures that in the steady state every arc is allowed to verify (3) after merging t_i and t_j. Therefore, after merging t_i and t_j the system throughput is preserved. It must be taken into account that the firing of t_i may produce a marking with negative values in some of its input arcs ${}^\bullet t_i$. Since such marking is not a valid initialization, the input transitions of the arcs with negative values must also be fired. These new firings may produce a new set of arcs with negative values, and then, more firings have to be carried out. Given that $\tilde{M}(a_i) = \tilde{M}(a_j)$ and $\tilde{M}(a_i') = 2 - \tilde{M}(a_i)$, where a_i' is the complementary arc of a_i, any of the cycles containing a_i' and a_j will have at least 2 tokens. This implies that the firing process to avoid negative markings (the lowest possible value is -2 since $M_0(a_i) - M_0(a_j) \leq 2$) will not fire transition t_j. Moreover, since each cycle has a positive number of tokens that is preserved by any firing sequence, it will not fire any cycle of transitions. Hence, such firing process will eventually finish, and will yield a marking M' in which all arcs have non-negative values and $M'(a_i) = M'(a_j)$. □