Synchronous Elastic Networks

Sava Krstić
Strategic CAD Labs, Intel Corporation
Hillsboro, Oregon, USA

Jordi Cortadella
Universitat Politècnica de Catalunya
Barcelona, Spain

Mike Kishinevsky, John O’Leary
Strategic CAD Labs, Intel Corporation
Hillsboro, Oregon, USA

Abstract—We formally define—at the stream transformer level—a class of synchronous circuits that tolerate any variability in the latency of their environment. We study behavioral properties of networks of such circuits and prove fundamental compositionality results. The paper contributes to bridging the gap between the theory of latency-insensitive systems and the correct implementation of efficient control structures for them.

I. INTRODUCTION

The conventional abstract model for a synchronous circuit is a machine that reads inputs and writes outputs at every cycle. The outputs at cycle \( i \) are produced according to a calculation that depends on the inputs at cycles \( 0, \ldots, i \). Computations and data transfers are assumed to take zero delay.

Latency-insensitive design by Carloni et al. [2] aims to relax this model by elasticizing the time dimension and so decoupling the cycles from the calculations of the circuit. It enables the design of circuits tolerant to any discrete variation (in the number of cycles) of the computation and communication delays. With this modular approach, the functionality of the system only depends on the functionality of its components and not on their timing characteristics.

The motivation for latency-insensitive design comes from the difficulties with timing and communication in nanoscale technologies. The number of cycles required to transmit data from a sender to a receiver is governed by the distance between them, and often cannot be accurately known until the chip layout is generated late in the design process. Traditional design approaches require fixing the communication latencies up front, and these are difficult to amend when layout information finally becomes available. Elastic circuits offer a solution to this problem. In addition, their modularity promises novel methods for microarchitectural design that can use variable-latency components and tolerate static and dynamic changes in communication latencies, while—unlike synchronous circuits—still employing standard synchronous design tools and methods.

Cortadella et al. [4] present a simple elastic protocol, called SELF (Synchronous Elastic Flow) and describe methods for efficient implementation of elastic systems and for conversion of regular synchronous designs into elastic form. Inspired by the original work on latency-insensitive design [2], SELF also differs from it in ways that render the theory developed in [2] hardly applicable.

In this paper we give theoretical foundations of SELF: a novel and arguably more practicable definition of elasticity, and the basic compositionality results. For space reasons, the proofs are omitted, but are available in the technical report [7].

A. Overview

Figure 1(a) depicts the timing behavior of a conventional synchronous adder that reads input and produces output data at every cycle (boxes represent cycles). In this adder, the \( i \)-th output value is produced at the \( i \)-th cycle. Figure 1(b) depicts a related behavior of an elastic adder—a synchronous circuit too—in which data transfer occurs in some cycles and not in others. We refer to the transferred data items as tokens and we say that idle cycles contain bubbles.

Put succinctly, elasticization decouples cycle count from token count. In a conventional synchronous circuit, the \( i \)-th token of a wire is transmitted at the \( i \)-th cycle, whereas in a synchronous elastic circuit the \( i \)-th token is transmitted at some cycle \( k \geq i \).

Turning a conventional synchronous adder into a synchronous elastic adder requires a communication discipline that differentiates idle from non-idle cycles (bubbles from tokens). In SELF, this is implemented by a pair of single-bit control wires: Valid and Stop. Every input or output wire \( Z \) in a synchronous component is associated to a channel in the elastic version of the same component. The channel is a triple of wires \( \langle Z, \text{valid}_Z, \text{stop}_Z \rangle \), with \( Z \) carrying the data and the other two wires implementing the control bits, as shown in Figure 2(b). A token is transferred on this channel when \( \text{valid}_Z \land \neg \text{stop}_Z \); the sender sends valid data and the receiver is ready to accept it; see Figure 4. Additional constraints that guarantee correct elastic behavior are given in Section III. There we define precisely the class of elastic circuits and what it means for a circuit \( A^e \) to be an elastization of a given circuit \( A \). In particular, our definition implies liveness: \( A^e \) produces infinite streams of tokens if its environment produces infinite streams of tokens at the input channels and is ready to accept infinite streams at the output channels.

Suppose \( \mathcal{N} \) is a network of standard (non-elastic) components, as in Figure 2(a). Suppose we then take elastizizations of

![Fig. 1. (a) Conventional synchronous adder, (b) Synchronous elastic adder.](image-url)
these standard components and join their channels accordingly, as in Figure 2(b), ignoring the buffer. Will the resulting network \( N^e \) be an elasticization of \( N \)? Will it be elastic at all? These fundamental questions are answered by Theorem 4 of Section IV, which is the main result of the paper. The answers are “yes”, provided a certain graph \( \Delta^e(N^e) \) associated with \( N^e \) is acyclic. This graph captures the information about paths inside elastic systems that contain no tokens—alogous to combinational paths in ordinary systems. Importantly, \( \Delta^e(N^e) \) can be constructed using only local information (the “sequentiality interfaces”) of the individual elastic components.

Since elastic networks tolerate any variability in the latency of the components, empty FIFO buffers can be inserted in any channel, as shown in Figure 2(b), without changing the functional behavior of the network. This practically important fact is proved as a consequence of Theorem 4.

Synchronous circuits are modeled in this paper as stream transformers, called machines. This well-known technique (see [8] and references therein) appears to be quite underdeveloped. Our rather lengthy preliminary Section II elaborates the necessary theory of networks of machines, culminating with a surprisingly novel combinational loop theorem (Theorem 1). Figure 3 illustrates Theorem 1 and, by analogy, Theorem 4 as well. It relies on the formalization of the notion of combinational dependence at the level of input-output wire pairs. Each input-output pair of a machine is either sequential or not, and the set of sequential pairs provides a machine’s “sequentiality interface”. When several machines are put together into a network \( N \), their sequentiality interfaces define the graph \( \Delta(N) \), the acyclicity of which is a test for the network to be a legitimate machine itself.

Elasticizations of ordinary circuits are not uniquely defined. On the other hand, for every elastic machine \( A \) there is a unique standard machine, denoted \( A^s \), that corresponds to it. We do not discuss any specific elasticization procedures in this paper, but state our results in the form that only involves elastic machines and their unique standard counterparts. This makes the results applicable to multiple elasticization procedures.

### B. Related Work

Carloni et al. [2] pioneered a theory of latency-insensitive circuits based on their notion of patient processes. Patient processes are defined at a high level of abstraction that models communication on a channel only by “token or bubble”, leaving implementation protocol(s) unspecified. In the companion paper [3], Carloni et al. give an incomplete description of an implementation protocol. Assuming our recovery of that protocol (let us call it LID) is accurate, its transfer condition is more complex than that of SELF (Figure 4) and consequently LID requires significantly more complex implementation. For example, conversion of a regular design into LID form needs a wrapper or registers around every module, increasing the latency of each module’s computation by two cycles—a penalty that is not required in the SELF elasticization. There might also be practical challenges in interfacing a LID system with an existing non-LID module, requiring the latter to generate stop signals with complex semantics.

We emphasize that the limitations of LID implementations are not inherent to the concept of patient processes. Regarding latency properties, they do not seem to be more limited than elastic systems. Still, it turns out that patient processes are not general enough to model elastic systems as we define them in Section III. This we prove in Section V where patient processes and elastic systems are compared as alternative formalizations of latency-insensitive circuits.

Suhaib et al. [12] revisited and generalized Carloni’s elasticization procedure, validating its correctness by a simulation method based on model checking.

Lee et al. [9] study causality interfaces (pairwise input-output dependencies) and are “interested in existence and uniqueness of the behavior of feedback composition”, but do not go as far as deriving a combinational loop theorem.

In their work on design of interlock pipelines [6], Jacobson et al. use a protocol equivalent to SELF, without explicitly...
specifying it.

Manohar and Martin discuss “slack elasticity” of asynchronous implementations in [10]. Their slack elasticity conditions relate to the structure of choices in the asynchronous specification. Unlike [10], in the current paper we deal with synchronous systems and we take a black box view of their control—no information about the control flow (and hence on the structure of choices) is ever used. Instead the connectivity information corresponding to the system data-flow is used for elasticization. Conservatively ignoring control flow may lead to a performance penalty, but simplifies the translation to an elastic system.

II. CIRCUITS AS STREAM FUNCTIONS

In this section we introduce machines as a mathematical abstraction of circuits without combinational cycles. For simplicity, this abstraction implicitly assumes that all sequential elements inside the circuit are initialized. Extending to partially initialized systems appears to be trivial. While there is a large body of work studying circuits or equivalent objects with good (e.g. constructive [1]) combinational cycles and their composition (e.g. [5]), we deliberately restrict consideration to the fully acyclic objects, since neither logic synthesis nor timing analysis can properly treat circuits with combinational cycles.

Most of the effort in this section goes into establishing modularity conditions guaranteeing that a system obtained as a network of machines (the feedback construction in particular) is a machine itself.

A. Streams

A stream over A is an infinite sequence whose elements belong to the set A. The first element of a stream a is referred to by a[0], the second by a[1], etc. For example, the equation

\[ a[i] = 3i + 1 \]

describes the stream (1, 4, 7, ...).

The set of all streams will be denoted \( A^\infty \). Occasionally we will need to consider finite sequences too; the set of all, finite or infinite, sequences over A is denoted \( A^\omega \).

We will write \( a \sim_k b \) to indicate that the streams a and b have a common prefix of length k. The equivalence relations \( \sim_0, \sim_1, \sim_2, \ldots \) are progressively finer and have trivial intersection. Thus, to prove two sequences a and b are equal, it suffices to show \( a \sim_k b \) holds for every k. Note also that \( a \sim_0 b \) holds for every a and b.

We will use the equivalence relations \( \sim_k \) to express properties of systems and machines viewed as multivariate stream functions. All these properties will be derived from the following two basic properties of single-variable stream functions f: \( A^\infty \to B^\infty \).

**Lemma 1:** If f: \( A^\infty \to A^\infty \) is contractive, then it has a unique fixpoint.

**Remark.** One can define the distance \( d(a, b) \) between sequences a and b to be \( 1/2^k \), where k is the length of the largest common prefix of a and b. This gives the sets \( A^\infty \) and \( A^\omega \) the structure of complete metric spaces and Lemma 1 is an instance of Banach Fixed Point Theorem. See the review paper [8] for more details and references about the metric semantics of systems and [13] for “diadic arithmetic of circuits”. We choose not to use the metric space terminology in this paper since all “metric reasoning” we need can be as easily done with equivalence relations \( \sim_k \) instead. See [11] for principles of reasoning with such “converging equivalence relations” in more general contexts.

B. Systems

Suppose W is a set of typed wires; all we know about an individual wire w is a set \( \text{type}(w) \) associated to it. A W-behavior is a function \( \sigma \) that associates a stream \( \sigma.w \in \text{type}(w)^\infty \) to each wire w ∈ W. The set of all W-behaviors will be denoted \( \llbracket W \rrbracket \). Slightly abusing the notation, we will also write \( \llbracket w \rrbracket \) for the set \( \text{type}(w)^\infty \). Notice that the equivalence relations \( \sim_k \) extend naturally from streams to behaviors:

\[ \sigma \sim_k \sigma' \iff \forall w \in W. \sigma.w \sim_k \sigma'.w \]

Notice also that a W-behavior \( \sigma \) can be seen as a single stream \( (\sigma[0], \sigma[1], \ldots) \) of W-states, where a state is an assignment of a value in \( \text{type}(w) \) to each wire w.

**Definition 1:** A W-system is a subset of \( \llbracket W \rrbracket \).

**Example.** A circuit that at each clock cycle receives an integer as input and returns the sum of all previously received inputs is described by the W-system \( S \), where W consists of two wires u, v of type Z, and \( S = \{ (u, v) | u+v = 0 \} \). This gives the sets \( \llbracket u \rrbracket \) and \( \llbracket v \rrbracket \) such that (u[0] = 0 and v[n] = a[0] + ... + a[n-1] for n > 0. Each stream pair (a, b) represents a behavior \( \sigma \) such that \( \sigma.u = a \) and \( \sigma.v = b \).

We will use wires as typed variables in formulas meant to describe system properties. The formulas are built using ordinary mathematical and logical notation, enhanced with temporal operators next, always, and eventually, denoted respectively by (□)*, G, F. As an illustration, the system \( S \) in the example above is characterized by the property \( v = 0 \land G(v > u) \). Also, one has \( S \models FG(u > 0) \models FG(u > 1000) \), where \( \models \) is used to denote that a formula is true of a system.

C. Operations on Systems

If W′ ⊆ W, there is an obvious projection map \( \sigma \mapsto \sigma \downarrow W' : \llbracket W \rrbracket \to \llbracket W' \rrbracket \). These projections are all one needs for the definition of the following two basic operations on systems.

**Definition 2:** (a) If S is a W-system and W′ ⊆ W, then hiding W′ in S produces a \( (W' - W') \)-system \( \text{hide}_{W'}(S) \) defined by

\[ \tau \in \text{hide}_{W'}(S) \iff \exists \sigma \in S. \tau = \sigma \downarrow (W - W') \]
(b) The composition of a $W_1$-system $S_1$ and a $W_2$-system $S_2$ is a $(W_1 \cup W_2)$-system $S_1 \sqcup S_2$ defined by

$$\sigma \in S_1 \sqcup S_2 \iff \sigma \upharpoonright W_1 \in S_1 \land \sigma \upharpoonright W_2 \in S_2.$$  

If $W$ and $W'$ are disjoint wire sets, $\sigma \in \llbracket W \rrbracket$, and $\tau \in \llbracket W' \rrbracket$, then there is a unique behavior $\vartheta \in \llbracket W \cup W' \rrbracket$ such that $\sigma = \vartheta \downharpoonright W$ and $\tau = \vartheta \downharpoonright W'$. This “product” of behaviors will be written as $\vartheta = \sigma \times \tau$. (If $W$ is the empty set, then $\llbracket W \rrbracket$ has one element—a “trivial behavior” that is also a multiplicative unit for the product operation $\times$.) We will also use the notation $[u \mapsto a, v \mapsto b, \ldots]$ for the $\{u, v, \ldots\}$-behavior $\sigma$ such that $\sigma.u = a$, $\sigma.v = b$, etc.

Hiding and composition suffice to define complex networks of systems. To model identification of wires, we use simple connection systems: by definition, $\text{Conn}(u, v)$ is the $\{u, v\}$-system consisting of all behaviors $\sigma$ such that $\sigma.u = \sigma.v$.

Now if $S_1, \ldots, S_m$ are given systems and $u_1, \ldots, u_n, v_1, \ldots, v_n$ are some of their wires, the network obtained from these systems by identifying each wire $u_i$ with the corresponding wire $v_i$ (of equal type) is the system

$$\langle S_1, \ldots, S_m \mid u_1 = v_1, \ldots, u_n = v_n \rangle$$

defined as $\text{hide}_{\{u_1, \ldots, u_n, v_1, \ldots, v_n\}}(S)$, where

$$S = S_1 \sqcup \cdots \sqcup S_m \sqcup \text{Conn}(u_1, v_1) \sqcup \cdots \sqcup \text{Conn}(u_n, v_n).$$

The simplest case ($m = n = 1$) of networks is the construction

$$\langle S \mid u = v \rangle = \text{hide}_{\{u, v\}}(S \sqcup \text{Conn}(u, v)),$$

used for a feedback definition in Section II-E. A behavior $\sigma$ belongs to $\langle S \mid u = v \rangle$ if and only if $\sigma \upharpoonright \{u \mapsto a, v \mapsto a\} \in S$ for some $a \in \llbracket u \rrbracket$.

D. Machines

Suppose $I$ and $O$ are disjoint sets of wires, called inputs and outputs, correspondingly. By definition, an $(I, O)$-system is just an $(I \cup O)$-system. Consider the following properties of an $(I, O)$-system $S$.

- **Deterministic:**
  $$\forall \omega, \omega' \in S. \ \omega \downharpoonright I = \omega' \downharpoonright I \implies \omega \downharpoonright O = \omega' \downharpoonright O$$

- **Functional:**
  $$\forall \omega \in \llbracket I \rrbracket. \ \exists \tau \in \llbracket O \rrbracket. \ \sigma \times \tau \in S$$

- **Causal:**
  $$\forall \omega, \omega' \in S. \ \forall k \geq 0. \ \omega \downharpoonright I \sim_k \omega' \downharpoonright I \implies \omega \downharpoonright O \sim_k \omega' \downharpoonright O$$

Clearly, functionality implies determinism. Conversely, a deterministic system is functional if and only if it accepts all inputs. Note also that causality implies determinism: if $\omega \downharpoonright I = \omega' \downharpoonright I$, then $\omega \downharpoonright I \sim_k \omega' \downharpoonright I$ holds for every $k$, so $\omega \downharpoonright O \sim_k \omega' \downharpoonright O$ holds for every $k$ too, so $\omega \downharpoonright O = \omega' \downharpoonright O$.

**Definition 3:** An $(I, O)$-machine is an $(I, O)$-system that is both functional and causal.

A functional system $S$ uniquely determines and is determined by the function $F: \llbracket I \rrbracket \to \llbracket O \rrbracket$ such that $F(\sigma) = \tau$ holds if and only if $\sigma \times \tau \in S$. The causality condition for such $S$ can be also written as follows:

$$\forall \sigma, \sigma' \in \llbracket I \rrbracket. \ \forall k \geq 0. \ \sigma \sim_k \sigma' \implies F(\sigma) \sim_k F(\sigma').$$

The system in the example in Section II-B is a machine if we regard $u$ as an input wire and $v$ as an output wire. The same is true of the system $\text{Conn}(u, v)$: its associated function $F$ is the identity function.

E. Feedback on Machines

We will use the term feedback for the system $\langle S \mid u = v \rangle$ as mentioned in Section II-C when $S$ is a machine and the wires $u$ and $v$ of the same type are an input and output of $S$ respectively. Our concern now is to understand under what conditions the feedback produces a machine.

To fix the notation, assume $S$ is an $(I, O)$-machine given by $F: \llbracket I \rrbracket \to \llbracket O \rrbracket$, with wires $u \in I, v \in O$ of the same type $A$. By the note at the end of Section II-C, we have that for every $\omega \in \llbracket I \setminus \{u\} \rrbracket$ and $\tau \in \llbracket O \setminus \{v\} \rrbracket$,

$$\sigma \times \tau \in \langle S \mid u = v \rangle$$

if and only if

$$\exists a \in A^\infty. \ F(\sigma \times [u \mapsto a]) = \tau \times [v \mapsto a],$$

so $\langle S \mid u = v \rangle$ is functional when the function $F_{uv}^\sigma: A^\infty \to A^\infty$ defined by $F_{uv}^\sigma(a) = F(\sigma \times [u \mapsto a])$ has a unique fixpoint. By Lemma 1, this is guaranteed if $F_{uv}^\sigma$ is contractive.

The following definition introduces the key concept of sequentiality that formalizes the intuitive notion that there is no combinational dependence of a given output wire on a given input wire. Sequentiality of the pair $(u, v)$ easily implies contractivity of $F_{uv}^\sigma$ for all $\sigma$.

**Definition 4:** The pair $(u, v)$ is sequential for $S$ if for every $\sigma, \sigma' \in \llbracket I \rrbracket$ and every $k \geq 0$

$$\sigma.u \sim_{k-1} \sigma'.u \land \forall x \in I \setminus \{u\}. \ (\sigma.x \sim_k \sigma'.x) \implies F(\sigma).v \sim_k F(\sigma').v$$

**Lemma 2 (Feedback):** If $(u, v)$ is a sequential input-output pair for a machine $S$, then the feedback system $\langle S \mid u = v \rangle$ is a machine too.

**Example:** Consider the system $S$ with $I = \{u, v\}$, $O = \{w, z\}$, specified by equations

$$w = u \oplus ((0) \# v) \quad z = v \oplus v,$$

where all wires have type $Z$, the symbol $\oplus$ denotes the componentwise sum of streams, and $\#$ denotes concatenation. Since $z$ does not depend on $u$, the pair $(u, z)$ is sequential. The pair $(v, w)$ is also sequential since to compute a prefix of $w$ it suffices to know a prefix of the same size of $u$ and a prefix of smaller size of $v$. The remaining two input-output pairs $(u, w)$ and $(v, z)$ are not sequential.

To find the machine $\langle S \mid v = w \rangle$, we need to solve the equation $v = u \oplus ((0) \# v)$ for $v$. For each $u = (a_0, a_1, a_2, \ldots)$, the equation has a unique solution $v = \hat{u} = (a_0, a_0 + a_1, a_0 + a_1 + a_2, \ldots)$. Substituting the solution into $z = v \oplus v$, we obtain
a description of \( \langle S | v = w \rangle \) by a single equation that relates its input and output: \( z = \bar{u} \oplus \bar{w} \). The other feedback \( \langle S | u = z \rangle \) is easier to calculate; it is given by \( u = v \oplus v \oplus \bar{(0 \# v)} \).

F. Networks of Machines and the Combinatorial Loop Theorem

Consider a network \( N = \langle S_1, \ldots, S_m | u_1 = v_1, \ldots, u_n = v_n \rangle \), where \( S_1, \ldots, S_m \) are machines with disjoint wire sets and the pairs \( (u_1, v_1), \ldots, (u_n, v_n) \) involve \( n \) distinct input wires \( u_i \) and \( n \) distinct output wires \( v_i \). (There is no assumption that \( u_i, v_i \) belong to the same machine \( S_i \).) Our goal is to understand under what conditions the system \( N \) is a machine.

Note that \( N = \langle S | u_1 = v_1, \ldots, u_n = v_n \rangle \), where \( S = S_1 \sqcup \cdots \sqcup S_m \). It is easy to check that an input-output pair \( (u, v) \) of \( S \) is sequential if either (1) \( (u, v) \) is sequential for some \( S_i \), or (2) \( u \) and \( v \) belong to different machines. Thus, the information about sequentiality of input-output pairs of the “parallel composition” machine \( S \) is readily available from the sequentiality information about the component machines \( S_i \), and our problem boils down to determining when a multiple feedback operation performed on a single machine results in a system that is itself a machine.

Simultaneous feedback specified by a set of two or more input-output pairs of a machine does not necessarily produce a machine even if all pairs involved are sequential. Indeed, in the example in Section II-E, we had a system \( S \) with two sequential pairs \( (u, z) \) and \( (v, w) \), but \( (u, z) \) ceases to be sequential for \( \langle S | v = w \rangle \). Indeed, if \( z \) and \( v \) are related by \( z = \bar{u} \oplus \bar{w} \), then knowing a prefix of length \( k \) of \( z \) requires knowing the prefix of the same length of \( w \); a shorter one would not suffice.

To ensure that a multiple feedback construction produces a machine, one needs to show that, in addition to the wire pairs to be identified, sufficiently many other input-output pairs are also sequential. A precise formulation for a double feedback is given by a version of the Bekić Lemma: for the system \( \langle S | u = w, v = z \rangle \) to be a machine, it suffices that three pairs of wires be sequential—\( (u, w) \), \( (u, z) \), and one of \( (u, z), (v, w) \). This non-trivial auxiliary result is needed for the proof of Theorem 1 below, and is a special case of it.

Given an \( (I, O) \)-machine \( S \), let its dependency graph \( \Delta(S) \) have the vertex set \( I \times O \) and directed edges that go from \( u \) to \( v \) for each pair \( (u, v) \in I \times O \) that is not sequential. For a network system \( N = \langle S_1, \ldots, S_m | u_1 = v_1, \ldots, u_n = v_n \rangle \), its graph \( \Delta(N) \) is then defined as the direct sum of graphs \( \Delta(S_1), \ldots, \Delta(S_m) \) with each vertex \( u_i \) (\( 1 \leq i \leq n \)) identified with the corresponding vertex \( v_i \) (Figure 3).

**Theorem 1 (Combinatorial Loop Theorem):** The network system \( N \) is a machine if the graph \( \Delta(N) \) is acyclic.

III. ELASTIC MACHINES

In this section we give the definition of elastic machines. Its four parts—input-output structure, persistence conditions, liveness conditions, and the transfer determinism condition—are covered by Definitions 5-8 below.

A. Input-output Structure, Channels, and Transfer

We assume that the set of wires is partitioned into data, valid, and stop wires, so that for each data wire \( X \) there exist associated wires \( \text{valid}_X \) and \( \text{stop}_X \) of boolean type. (In actual circuit implementations, valid\_X and stop\_X need not be physical wires; it suffices that they be appropriately encoded.)

**Definition 5:** Let \( I, O \) be disjoint sets of data wires. An \((I, O)\)-system is an \((I', O')\)-machine, where \( I' = I \cup \{ \text{valid}_X | X \in I \} \cup \{ \text{stop}_Y | Y \in O \} \) and \( O' = O \cup \{ \text{valid}_Y | Y \in O \} \cup \{ \text{stop}_X | X \in I \} \).

The triples \( \langle X, \text{valid}_X, \text{stop}_X \rangle \) (for \( X \in I \)) and \( \langle Y, \text{valid}_Y, \text{stop}_Y \rangle \) (for \( Y \in O \)) are to be thought of as elastic input and output channels of the system.

Let \( \text{transfer}_Z \) be a shorthand for \( \text{valid}_Z \land \neg \text{stop}_Z \) and say that \( \text{transfer} \) along \( Z \) occurs in a state \( s \) if \( s \models \text{transfer}_Z \).

Given a behavior \( \sigma = (\sigma[0], \sigma[1], \sigma[2], \ldots) \) of an \((I, O)\)-system and \( Z \in I \cup O \), let \( \sigma_Z \) be the sequence (perhaps finite!) obtained from \( \sigma.Z = (\sigma[0].Z, \sigma[1].Z, \sigma[2].Z, \ldots) \) by deleting all entries \( \sigma[i].Z \) such that transfer along \( Z \) does not occur in \( \sigma[i] \). The transfer behavior \( \sigma^T \) associated with \( \sigma \) is then defined by \( \sigma^T.Z = \sigma_Z. \) If all sequences \( \sigma_Z \) are infinite, then \( \sigma^T \) is an \((I \cup O)\)-behavior; in general, however, we only have \( \sigma_Z \in \text{type}(Z)^\omega \).

For each wire \( Z \) of an \((I, O)\)-system \( S \) we introduce an auxiliary transfer counter variable \( \text{tct}_Z \) of type \( Z \). The counters serve for expressing system properties related to transfer. By definition, \( \text{tct}_Z \) is equal to the number of states that precede the current state and in which transfer along \( Z \) has occurred. That is, for every behavior \( \sigma \) of \( S \), we have \( \sigma.\text{tct}_Z = (t_0, t_1, \ldots) \), where \( t_k \) is the number of indices \( i \) such that \( i < k \) and transfer along \( Z \) occurs in \( \sigma[i] \). Note that the sequence \( \sigma.\text{tct}_Z \) is non-decreasing and begins with \( t_0 = 0 \).

The notation \( \min.\text{tct}_S \), for any subset \( S \) of \( I \cup O \) will be used to denote the smallest of the numbers \( \text{tct}_Z \), where \( Z \in S \).

B. Definition of Elasticity

An elastic component, when ready to communicate over an output channel must remain ready until the transfer takes place.

**Definition 6:** The persistence conditions for an \((I, O)\)-system \( S \) are given by

\[
S \models G(\text{valid}_Y \land \text{stop}_Y \Rightarrow (\text{valid}_Y)^+ \land Y^+ = Y) \quad (1)
\]

for every \( Y \in O \).

The conjunct \( Y^+ = Y \) can be removed from (1) without affecting the definition of elastic machines (it follows from other conditions). The most useful consequence of persistence is the “handshake lemma”:

\[
S \models GF\text{valid}_Y \land GF\neg\text{stop}_Y \Rightarrow GF\text{transfer}_Y
\]

Liveness of an elastic component is expressed in terms of token count: if all input channels have seen \( k \) transfers and there is an output channel that has seen less, then the communication on output channels with the minimum amount of transfer must be eventually offered. The following definition formalizes this,
together with a similar commitment to eventual readiness on input channels. (See also Figure 5.)

**Definition 7:** The *liveness conditions* for an \([I, O]\)-system are given by

\[
S \models G \left( \min_{tctO} tct_Y \land \min_{tct} tct_Y \Rightarrow F \text{valid}_Y \right) (2)
\]

\[
S \models G \left( \min_{tctI, O} tct_X \Rightarrow F \neg \text{stop}_X \right) (3)
\]

for every \(Y \in O\) and every \(X \in I\).

In practice, elastic components will satisfy simpler (but stronger) liveness properties; e.g. remove \(\min_{tctO} tct_Y\) from (2) and replace \(\min_{tctI, O} tct_X\) with \(\min_{tctO} tct_Y\) in (3). However, a composition of such components, while satisfying (2) and (3), may not satisfy the stronger versions of these conditions.

Consider single-channel \([I, O]\)-systems satisfying the persistency and liveness conditions; an *elastic consumer* is a \([[Z], \emptyset]]\)-system \(C\) satisfying (4) below; similarly, an *elastic producer* is a \([\emptyset, [Z]]\)-system \(P\) satisfying (5) and (6).

\[
C \models GF \neg \text{stop}_Z \quad (4)
\]

\[
P \models G (\text{valid}_Z \land \neg \text{stop}_Z \Rightarrow (\text{valid}_Z)\) \quad (5)
\]

\[
P \models GF \text{valid}_Z \quad (6)
\]

Let \(C_2\) be the \([Z, \text{valid}_Z, \text{stop}_Z]\)-system characterized by condition (4)—the largest (in the sense of behavior inclusion) of the systems satisfying this condition. Similarly, let \(P_2\) be the \([Z, \text{valid}_Z, \text{stop}_Z]\)-system characterized by properties (5) and (6). Finally, define the \([I, O]\)-*elastic environment* to be the system

\[
\text{Env}_{I, O} = \bigcup_{X \in I} P_X \sqcup \bigcup_{Y \in O} C_Y.
\]

Note that \(\text{Env}_{I, O}\) is only a system; it is not functional and so is not a machine.

When a system satisfying the persistency and liveness conditions (1-3) is coupled with a matching elastic environment, the transfers on all data wires never comes to a stall:

**Lemma 3 (Liveness):** If \(S\) satisfies (1-3), then for every behavior \(\omega\) of \(S \sqcup \text{Env}_{I, O}\), all the component sequences of the transfer behavior \(\omega^\Sigma\) are infinite.

As an immediate consequence of Liveness Lemma, if \(S\) satisfies (1-3), then

\[
S^\Sigma = \{ \omega^\Sigma \mid \omega \in S \sqcup \text{Env}_{I, O} \}
\]

is a well-defined \([I, O]\)-system.

**Definition 8:** An \([I, O]\)-system \(S\) is an \([I, O]\)-*elastic machine* if it satisfies the properties (1-3) and the associated system \(S^\Sigma\) is deterministic.

The liveness conditions (2,3) are visibly related to causality at the transfer level: \(k\) transfers on the input channels imply \(k\) transfers on the output channels in the cooperating environment. Thus, it is not surprising (even though the proof is not obvious) that the determinism postulated in Definition 8 suffices to derive the causality of \(S^\Sigma\):

**Theorem 2:** If \(S\) is an \([I, O]\)-elastic machine, then \(S^\Sigma\) is an \((I, O)\)-machine.

In the situation of Definition 8, we say that \(S\) is an *elasticization* of \(S^\Sigma\) and that \(S^\Sigma\) is the transfer machine of \(S\).

**IV. Elastic Networks**

An *elastic network* \(N\) is given by a set of elastic machines \(S_1, \ldots, S_m\) with no shared wires, together with a set of channel pairs \((X_1, Y_1), \ldots, (X_n, Y_n)\), where the \(X_i\) are \(n\) distinct input channels and the \(Y_i\) are \(n\) distinct output channels. As a network of standard machines, the elastic network \(N\) is defined by

\[
N = \langle S_1, \ldots, S_m \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle.
\]

We will define a graph that encodes the sequentiality information about the network \(N\) and prove in Theorem 4 that acyclicity of that graph implies that \(N\) is an elastic machine and that \(N^\Sigma = \langle S_1^\Sigma, \ldots, S_m^\Sigma \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle\).

**A. Elastic Feedback**

**Elastic feedback** is a simple case of elastic network:

\[
\langle S \parallel P = Q \rangle = \langle S \mid P = Q, \text{valid}_P = \text{valid}_Q, \text{stop}_P = \text{stop}_Q \rangle.
\]

**Definition 9:** Suppose \(S\) is an elastic machine. An input-output channel pair \((P, Q)\) will be called *sequential* for \(S\) if

\[
S \models G \left( \min_{tctI, O} tct_Q = tct_Q \land \min_{tctI - P} tct_Q \Rightarrow F \text{valid}_Q \right) \quad (7)
\]

Condition (7) is a strengthening of the liveness condition (2) for channel \(Q\). It expresses a degree of independence of the output channel \(Q\) from the input channel \(P\); e.g., the first token at \(Q\) need not wait for the arrival of the first token at \(P\). This independence can be achieved in the system by storing some tokens inside, between these two channels. Note that (7) does not guarantee that connecting channels \(P\) and \(Q\) would not introduce ordinary combinational cycles. Therefore the acyclicity condition in the following theorem is required to ensure (by Theorem 1) that the elastic feedback, viewed as an ordinary network, is a machine.

**Theorem 3:** Let \(S\) be an elastic machine and \(F\) the elastic feedback system \(\langle S \parallel P = Q \rangle\). If the channel pair \((P, Q)\) is sequential for \(S\), then: (a) the wire pair \((P, Q)\) is sequential for \(S^\Sigma\); if, in addition, \(\Delta(F)\) is acyclic, then: (b) \(F\) is an elastic machine, and (c) \(F^\Sigma = \langle S^\Sigma \mid P = Q \rangle\).
B. Main Theorems

Sequentiality of two channel pairs \((P, Q), (P', Q')\) of an elastic machine does not imply their “simultaneous sequentiality”

\[
S \models G \left( \min_{\delta \subseteq \delta^0} \text{tct}_{Y} \land \min_{\delta \subseteq \delta^1} \text{tct}_{Y} \rightarrow \text{tct}_{Q} \Rightarrow \text{F valid}_{Q} \right).
\]

This deviates from the situation with ordinary machines, where the analogous property holds and is instrumental in the proof of Combinational Loop Theorem.

To justify multiple feedback on elastic machines, we have thus to postulate that simultaneous sequentiality is true where required. Specifically, we demand that elastic machines come with simultaneous sequentiality information: If \(S\) is an \([I, O]\)-elastic machine, then for every \(Y \subseteq O\) a set \(\delta(Y) \subseteq I\) is given so that

\[
S \models G \left( \min_{\delta \subseteq \delta^0} \text{tct}_{Y} \land \min_{\delta \subseteq \delta^1} \text{tct}_{Y} \rightarrow \text{F valid}_{Y} \right).
\]

Note that if \(P \subseteq \delta(Q)\), then the pair \((P, Q)\) is sequential, but the converse is not implied. A function \(\delta: O \rightarrow 2^I\) with the property (8) will be called a sequentiality interface for \(S\).

For an \([I, O]\)-elastic machine \(S\) with a sequentiality interface \(\delta\), we define \(\delta^0(S, \delta)\) to be the graph with the vertex set \(I \cup O\) and directed edges \((X, Y)\) where \(X \notin \delta(Y)\). By Theorem 3(a), \(\delta^0(S, \delta)\) contains \(\delta(S^0)\) as a subgraph.

Given an elastic network \(N = \langle S_1, \ldots, S_n \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle\), where each \(S_i\) comes equipped with a sequentiality interface \(\delta_i\), its graph \(\delta^0(N)\) is by definition the direct sum of graphs \(\delta^0(S_1, \delta_1), \ldots, \delta^0(S_n, \delta_n)\) with each vertex \(X_i (1 \leq i \leq n)\) identified with the corresponding vertex \(Y_i\).

Theorem 4: If the graphs \(\delta^0(N)\) and \(\delta^0(N')\) are acyclic, then the network system \(N\) is an elastic machine, the corresponding non-elastic system \(N' = \langle S'_1, \ldots, S'_n \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle\) is a machine, and \(N^\tau = N\).

As in Theorem 3, acyclicity of \(\delta^0(N)\) is needed to ensure (by Theorem 1) that \(N'\) defines a machine.

Elasticization procedures (e.g. [4]) will typically produce elastic components with enough sequential input-output wire pairs, so that \(\delta^0(N)\) will be acyclic as soon as \(\delta^0(N')\) is acyclic.

Note, however, that cycles in \(\delta^0(N')\) need not correspond to combinational cycles in \(N\) seen as an ordinary network, since empty buffers with sequential elements cutting the combinational feedbacks may be inserted into \(N\). Even though non-combinational in the ordinary sense, these cycles contain no tokens and therefore no progress along them can be made.

Theorem 4 implies that insertion of empty elastic buffers does not affect the basic functionality of an elastic network, as illustrated in Figure 2(b).

Definition 10: An empty elastic buffer is an elastic machine \(S\) such that \(S^\tau = \text{Conn}(X, Y)\) for some \(X, Y\).

Theorem 5 (Buffer Insertion Theorem): Suppose \(B\) is an empty elastic buffer with channels \(X, Y\). Let \(N' = \langle S_1, \ldots, S_n \mid X_1 = Y_1, \ldots, X_n = Y_n \rangle\) and \(M = \langle B, S_1, \ldots, S_n \mid X = Y_1, X_1 = Y, X_2 = Y_2, \ldots, X_n = Y_n \rangle\).

If \(\Delta(N), \Delta(M), \) and \(\Delta^\tau(N)\) are acyclic, then \(M\) is an elastic machine, and \(M^\tau = N^\tau\).

The precise relationship between graphs \(\Delta(M)\) and \(\Delta(N)\) can be easily described. In practice they are at the same time acyclic or not, as a consequence of sequentiality of sufficiently many input-output wire pairs of \(B\).

V. ELASTIC VS. PATIENT SYSTEMS

Elastic machines and patient processes of [2] provide two formalizations of the intuitive concept of latency-insensitive circuits. In this section we address their connections and differences. We begin with an overview of [2], using a minimalistic approach and terminology that differs from the original. We believe, however, that Definition 11 below matches the original definition accurately in most important aspects.

A. Patient Systems

The notation \(A^*\) is for the set of finite sequences over \(A\). A finitary \(W\)-system, by definition, is a set of behaviors \(\sigma\) such that \(\sigma.w\) is a finite sequence for every \(w \in W\).

A stalling stream over \(A\) is a stream over \(A \cup \{\delta\}\). We will refer to \(\delta\) as the bubble and to elements of \(A\) as tokens. We will consider only stalling streams that contain finitely many tokens. If \(a\) is such a stream, let \(\pi \in A^*\) denote the sequence over \(A\) obtained by dropping all bubbles from \(a\). Clearly, \(a\) is determined by \(\pi\) and the sequence \(\delta(a) \in \mathbb{N}^*\) of lengths of bubble sequences between consecutive tokens of \(a\). For example, if

\[
a = (\square, \square, 7, \square, 4, 5, \square, \square, 8, \ldots)
\]

we have \(\pi = (7, 4, 5, 8, \ldots)\) and \(\delta(a) = (2, 1, 0, 3, \ldots)\). Two stalling streams \(a, b\) are latency equivalent, written \(a \equiv b\), when \(\pi = \bar{b}\). Note that \(a \equiv \pi\).

By definition, a stalling \(W\)-system is a set of behaviors \(\sigma\) such that for every \(w \in W\), \(\sigma.w\) is a stalling stream over \(\text{type}(w)\). Latency equivalence extends to \(W\)-behaviors and \(W\)-systems: \(\sigma \equiv \tau\) iff \(\sigma.w \equiv \tau.w\) holds for every \(w \in W\); \(\sigma \equiv \tau\) iff for every \(\sigma \in S\) \((\sigma \in S')\) there exists \(\tau \in S'\) \((\tau \in S)\) such that \(\sigma \equiv \tau\).

A stalling \(W\)-system \(S\) determines a standard finitary \(W\)-system \(S^\tau = \{\sigma \mid \sigma \in S\}\), where \(\sigma\) is given by \(\sigma.w = \sigma.w\) \((\forall w \in W)\). Clearly, \(S^\tau \equiv S\).

Stalling the \(k\)-th token of \(a\) by \(d\) steps produces a latency equivalent stream that will be denoted \(\text{stall}(a, k, d)\). Omitting the easy definition, we give an example: if \(a\) is as in (9), then

\[
\text{stall}(a, 1, 3) = (\square, \square, 7, \square, \square, \square, 4, 5, \square, \square, 8, \ldots)
\]

Definition 11: Let \(\prec\) be a well-founded order\(^1\) on \(W\) and let \(D > 0\). A patient \(W\)-system (relative to \(\prec\) and \(D\)) is a

\(^1\)Introduction of a well-founded ordering of wires is motivated in [2] with the purpose of modeling combinational dependencies, but such dependencies in patient systems are not discussed in any detail. Moreover, the ordering of wires is implicitly assumed to be total in [2], which is somewhat unnatural. For instance, when constructing a patient adder with inputs \(u, v\) and output \(w\), one has two ordering choices: \(u \prec_1 v \prec_1 w\) and \(v \prec_2 u \prec_2 w\). It is not clear that a patient adder in the \(\prec_1\)-sense will be patient in the \(\prec_2\)-sense too.
stalling system \( P \) such that for every \( \sigma \in P \), every \( u \in W \), and every \( k \geq 0 \) there exists \( \sigma' \in P \) such that
\[
\text{(Pat-1)} \quad \sigma'.u = \text{stall}(\sigma.u, k, 1)
\]
and for every \( v \neq u \) there exists \( d_v \leq D \) such that
\[
\text{(Pat-2)} \quad \sigma'.v = \begin{cases} 
\text{stall}(\sigma.v, k, d_v) & \text{if } u < v \\
\text{stall}(\sigma.v, k + 1, d_v) & \text{otherwise}
\end{cases}
\]

The main results of [2] can now be summarized:
1) a theorem saying that the composition of patient systems (with the same \( W \), \( \prec \), and \( D \)) is a patient system;
2) the definition and analysis of patient buffers, i.e. patient systems \( B \) such that \( B^T = \text{Conn}^{\text{fin}}(u, v) \)—the finitary connection system;
3) a general construction that, for a given finitary system \( M \) without combinational dependencies (model of a Moore machine), produces a patient system \( P \) such that \( P \cong M \).

B. Comparison

The formalization given by patient systems is at a higher level of abstraction. While elastic machines deal explicitly with handshaking signals between communicating systems, patient systems communicate purely in the token/bubble language. Given an elastic (as defined in Section III) \( (I, O) \)-system \( E \), the corresponding stalling \( (I \cup O) \)-system \( E^\Box \) is obtained by projecting the finite-transfer behaviors of \( E \) to data wires and replacing data items on each wire with \( \square \) at all cycles where transfer along that wire does not occur. Precisely, let \( E^\Box \) be the subset of \( E \) consisting of all behaviors \( \omega \) such that \( \omega^T.Z \) is finite for all channels \( Z \).

Then, given \( \omega \in E^\Box \), we define a stalling \( (I \cup O) \)-behavior \( \omega^\Box \) by
\[
(\omega^\Box.Z)[i] = \begin{cases} 
(\omega.Z)[i] \text{ if } (\omega.\text{valid}_Z)[i] \land \neg(\omega.\text{stop}_Z)[i] \\
\square & \text{otherwise}
\end{cases}
\]
and finally we define the stalling system \( E^\Box \) as the set of all such behaviors \( \omega^\Box \). Clearly, the system \( E^\Box \) is the finitary version of the standard machine \( E^T \).

Now we can address some questions pertinent to the comparison of patient processes vs. elastic machines.

Are patient processes more general? The answer is “no” because there exist elastic machines \( E \) such that \( E^\Box \) is not patient. To see this, consider an elastic machine \( E \) that starts offering new valid outputs on channel \( u \) only on even cycles. (The existence of such elastic machines is obvious.) Observe that \( \sigma.u = (\square, 7, 9, \ldots) \) is possible for some behavior \( \sigma \) of \( E^\Box \) (token 7, even though transmitted on cycle 1 was first offered on cycle 0). Then \( \text{stall}(\sigma.u, 0, 1) = (\square, \square, 7, 9, \ldots) \) must also be part of a behavior of \( E^\Box \), by condition (Pat-1) of Definition 11. This implies that token 9 is first offered on cycle 3, contrary to our assumption.

The above example can be viewed as an indication that the condition (Pat-1) is too restrictive. It would be interesting to see if an appropriate modification of (Pat-1) results in a definition of patient processes that captures elastic machines.

\[
\text{One can prove that } E \text{ is the set of all limits of behaviors of } E^\Box \text{ and so } E \text{ is determined by } E^\Box.
\]

Are elastic machines more general? The answer is an easy “no” since, for example, the set of all possible stalling \( W \)-behaviors is a patient system in the sense of Definition 11. However, if one adds to Definition 11 a reasonable requirement that a patient system be a machine, the answer is not immediately clear.

Which formalization is easier to use? Without offering a definitive answer, we would argue that verifying that a low-level design (RTL, say) implements an elastic machine would be easier than verifying that it implements a patient system. The bottom line is that the conditions for a system to be an elastic machine are expressible as temporal properties of suitably constructed infinite-state models. This is not obvious for the determinism condition for \( S^T \) in Definition 8, but can be done by replacing determinism with causality and introducing auxiliary variables for sequences of transferred values over channels. Even though (e.g., because of infinite counters involved) these conditions are not directly checkable by the existing model checking technology, there are palpable opportunities to find manageable stronger conditions that taken together imply elasticity (e.g., postulating a limit on the token count differences between channels eliminates the need for infinite counters). On the other hand, the definition of a patient system, being of the form “for every behavior \( \sigma \), there exists a behavior \( \sigma' \) such that …”, appears to us to be intrinsically more complex. Our only positive conclusion, however, is that the mechanical checking of either of the definitions is an open problem deserving further study.

VI. CONCLUSION

We have presented a theory of elastic machines that gives an easy-to-check condition for the compositional theorem of the form “an elasticization of a network of ordinary components is equivalent to the network of components’ elasticizations”. Verification of a particular implementation is reduced to proving that conditions of Definition 8 are satisfied for all elastic components used, and that the graph \( \Delta^a(\mathcal{N}^\infty) \) is acyclic for every network \( \mathcal{N} \) such that the elasticization is applied. While the definition of the graphs \( \Delta^a \) may appear complex because of the sequentaility interfaces involved, it should be noted that the elasticization procedures, e.g. [4], are reasonably expected to completely preserve sequentiality: a channel \( P \) belongs to \( \delta(Q) \) if the wire-pair \( (P, Q) \) is sequential in the original non-elastic machine. This ensures \( \Delta^a(\mathcal{N}^\infty) = \Delta(\mathcal{N}) \) and so testing for sequentiality is done at the level of ordinary networks.

Future work will be focused on proving correctness of particular elasticization methods, on techniques for mechanical verification of elasticity, and on extending the theory to more advanced protocols.

Acknowledgments: Luca Carloni clarified some details of [2]. Ken McMillan pointed out several inaccuracies in a previous version of the paper and further clarified [2] for us. Gerard Berry, Ching-Tsun Chou, John Harrison, and the anonymous reviewers provided useful remarks. We are grateful for all the help we received.
REFERENCES


