Robustness of Sequential Circuits

Laurent Doyen, LSV, ENS Cachan, France
Thomas A. Henzinger, IST Austria, Austria
Axel Legay, INRIA-Irisa, France
Dejan Ničković, IST Austria, Austria

***Abstract*—Digital components play a central role in the design of complex embedded systems. These components are interconnected with other, possibly analog, devices and the physical environment. This environment cannot be entirely captured and can provide inaccurate input data to the component. It is thus important for digital components to have a robust behavior, i.e. the presence of a small change in the input sequences should not result in a drastic change in the output sequences.**

In this paper, we study a notion of robustness for sequential circuits. However, since sequential circuits may have parts that are naturally discontinuous (e.g., digital controllers with switching behavior), we need a flexible framework that accommodates this fact and leaves discontinuous parts of the circuit out from the robustness analysis. As a consequence, we consider sequential circuits that have their input variables partitioned into two disjoint sets: control and disturbance variables. Our contributions are (1) a definition of robustness for sequential circuits as a form of continuity with respect to disturbance variables, (2) the characterization of the exact class of sequential circuits that are robust according to our definition, (3) an algorithm to decide whether a sequential circuit is robust or not.

I. Introduction

In embedded systems, digital components play a central role in the overall system. The external environment of digital components can be other digital or analog components, software, or the actual physical world. This is in contrast with the traditional (computer science) view of digital components, where the system is often considered to be either closed or to interact with an idealized environment that can be accurately modeled by a finite-state machine. In embedded systems, the environment cannot be always captured and precisely described in such a model. It follows that the input assumptions often remain inaccurate or incomplete. Additionally, the input data provided to the component by its environment can contain errors and imprecisions, either due to external perturbations, to the poor accuracy of sensors, or to unpredictable delays in communication links. As a consequence, a well-designed system needs to deal with such unexpected inputs. Therefore, it is important that the output behavior of such systems remains robust in presence of small disturbances in the input sequence. For critical applications, such as the flight controllers, it is essential to design systems that guarantee robust behavior even when the input is correct, to guarantee smooth control action. The design of robust components has been identified as one of the main challenges in the design of embedded systems [Hen08].

Traditionally, robustness and stability of systems are mainly studied in a continuous setting. Robustness of continuous systems is usually expressed as a form of *uniform continuity*. A system is uniformly continuous if for every positive real ϵ, there exists a positive real δ, such that any change smaller than δ in the input results in a change smaller than ϵ in the output. In control theory, it is also common to distinguish between *input* and *disturbance* variables, and to study robustness only with respect to disturbance variables.

In this paper, we extend the study of robustness to *sequential circuits*, i.e., discrete input/output finite-state systems composed of logic gates and delay elements interconnected by wires. Such circuits are a standard model for the design of complex digital hardware and finite-state transducers in general. Finite-state transducers have been intensively studied in the field of computer science and lead to some of the most fundamental results in the field. Adapting the techniques for robustness analysis from the continuous to the discrete setting is not straightforward and, however standard, the notion of uniform continuity is not directly applicable to digital systems:

1) Digital systems can be naturally discontinuous. Moreover, the discontinuity may often be a wanted property for some sub-components in a digital systems. This is true, in particular, if the component implements a discrete controller with switching functionality. On the other hand, we would like to guarantee that other components, such as critical systems, exhibit continuous behavior.
2) Another difference with respect to the continuous setting is that the changes in inputs and outputs of a digital system are more naturally quantified using integers instead of reals. The definition of uniform continuity combined with distance functions taking integer values results in every discrete system being uniformly continuous, simply by choosing δ smaller than 1.

The main contributions of this paper are twofold. First, we propose a new framework for studying robustness of sequential circuits based on simple and clean concepts that address the above-mentioned problems resulting from the underlying discrete setting. Inspired by the notion of robustness in control theory, we first make a distinction between two types of *input* alphabets: *control* and *disturbance* alphabets. Control alphabet encodes the control actions and disturbance alphabet encodes the environment actions. The distinction between control and disturbance alphabets is essential for providing a flexible framework for robustness analysis of sequential circuits. In particular, it allows to identify parts of the circuit that are wanted to be discontinuous and to eliminate them from the robustness analysis. This is because control actions are expected to cause discontinuities (e.g., in controllers that exhibit switching behavior), and as a consequence, we want to have a framework that only considers the effect of small changes in the disturbance actions. Due to their inherent discontinuous nature, the correctness of control inputs needs to be provided by using other methods that are outside of the scope of this paper, such as the error codes or input replication.

The main challenge for defining robustness as a form of continuity is to choose an appropriate notion of distance in the discrete setting. We identify such a metric, that we call *common suffix distance*. The common suffix distance gives the last position in which two sequences mismatch.

Finally, we define what we call Σ_D*-robustness* of sequential circuits as a form of continuity with respect to disturbance actions in the common suffix topology. Intuitively, Σ_D-robustness can be viewed as bounded propagation of changes in disturbance inputs: a finite number of changes in the disturbance values results in a bounded number of changes in the computed outputs.

We illustrate our notion of Σ_D-robustness for sequential circuits with the example of a 2-bit half-adder, which is depicted in Figure I (a). A 2-bit half-adder implements a simple arithmetic function that takes as inputs two boolean data values d_1 and d_2 and computes their *sum* and *carry* values. We model d_1 and d_2 as disturbance variables. This circuit is clearly Σ_D-robust with respect to d_1 and d_2 because at any time t, the outputs sum and $carry$ depend only on values of d_1 and d_2 at t. It follows that the effect of a change in input data (d_1 and d_2) at some point in time, is only limited to the computation of output values at that time without being propagated further in the future. In Figure I (b), the half-adder is connected to a small controller that operates as follows: (1) as long as ctr remains low, the half-adder is inactive and sum and $carry$ remain low, (2) when ctr is set to high for the first time, the half-adder is irreversibly activated and from then on, the controlled circuit copies the output values of the half-adder. Switching ctr to high has the qualitative effect on the behavior of the circuit. For instance, a sequence 0^ω of ctr values causes the half-adder to remains inactive forever. As a consequence, the output values of sum and $carry$ do not depend on d_1 and d_2 and always remain low by default. For a sequence $1 \cdot 0^\omega$, that differs from 0^ω only in the value of ctr at the first position, the output behavior of the circuit changes drastically. Now, the values of sum and $carry$ represent the result of the half-adder computation on inputs d_1 and d_2. In this example, the controller clearly causes discontinuities in the circuit behavior. However, ctr is a control variable and as such does not affect the Σ_D-robustness analysis, and the circuit does remain Σ_D-robust with respect to d_1 and d_2. This example is a simplified version of a more detailed example described in Section VI.

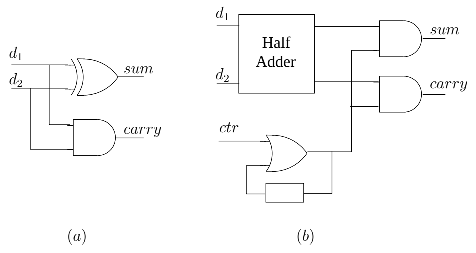


Figure 1. (a) 2-bit half-adder (b) Controlled 2-bit half-adder

Our definition of Σ_D-robustness relates pairs of input sequences that a sequential circuit consumes to the pairs of output sequences that it generates, without explicit reference to the underlying structure of the circuit. We give an alternative structure-less characterization of Σ_D-robust sequential circuits that is expressed in terms of the internal memory requirements for the storage of previous inputs in order to be able to compute the output. Given a sequential circuit with a partition of control Σ_C and disturbance Σ_D input alphabets, we define the class of circuits that have *finite disturbance horizon*. A circuit has finite disturbance horizon if it has a bound b such that the output at any time is computed as a function of, possibly, all previous control inputs, but only up to b previous disturbance inputs. We show that Σ_D-robust sequential circuits coincide exactly with circuits that have finite disturbance horizon.

Another contribution of the paper is the characterization of the class of Σ_D-robust sequential circuits in the form of their structural properties. This result is obtained by studying Σ_D-robustness for Mealy machines – a model that corresponds to the class of deterministic and synchronous finite-state transducers. We define a class of Σ_D*-synchronized* Mealy machines and show that is equivalent to corresponding Σ_D-robust sequential circuits. Σ_D-synchronization is an explicit state-based property of Mealy machines. Intuitively, a Mealy machine is Σ_D-synchronized if after reading two finite words of the same length with the same control, but possibly differing disturbance components, the machine is guaranteed to reach a “reset” state after reading any sufficiently long (continuation) sequence.

The above characterization of Σ_D-robust Mealy machines is operational and results in an effective algorithm for checking the Σ_D-robustness of arbitrary Mealy machines. The time complexity of the algorithm is quadratic in the number of states of the Mealy machine and the size of its disturbance alphabet and linear in the size of its control alphabet.

Related work: Robustness of systems has been studied and formalized in many different ways. For example, in [CB02], [KC04] robustness of hybrid systems (mixing discrete and continuous behaviors) is studied as a form of continuity in topologies induced by (extensions of) the Skorokhod distance. The immunity of finite state automata to noise from the external environment has been studied in [DM94], but this study is done in a probabilistic setting. Robustness of finite-state systems has been studied much less in the area of computer science. The work in [BGHJ09] focuses on the synthesis of robust discrete controllers satisfying an extended temporal logic specification with quantitative (real-valued) constraints, which results in a different model of robustness. In the context of real-time systems, robustness has been studied in [GHJ97]. This work differs from ours in that (1) the authors study acceptors (automata) instead of transducers and (2) pairs of timed sequences are compared using the Hausdorff distance with the limitation that the perturbations are allowed only in the time delays between events, while the discrete part has to match exactly. Robustness is also closely related to other research areas such as fault tolerance and error resiliation [SLM09]. In [Mal93], the authors consider combinatorial circuits with cycles, and study conditions under which such circuits can be expressed as an equivalent acyclic circuit. This work can be related to robustness questions since the presence of feedback loops in a circuit is a necessary condition for generating non-robust behavior. Finally, we observe that the problem of robustness can be seen as a dual of the problem of coverage, where one expects that an input change does result in drastic output change [KLS08].

II. Preliminaries

We first recall the classical definitions of distance and metric. A *distance* on a set S is a function $d : S \times S \to \mathbb{R} \cup \{\infty\}$. In general, one considers metrics that extend the definition of distance with some natural properties. The distance d is a *metric* if for all $x, y, z \in S$ (1) $d(x, y) \geq 0$, (2) $d(x, y) = 0$ if and only if $x = y$, (3) $d(x, y) = d(y, x)$, and (4) $d(x, y) \leq d(x, z) + d(z, y)$.

Let $\Sigma = \Sigma_C \times \Sigma_D$ and Γ be (2- and 1-dimensional) finite alphabets. A *finite sequence* $\hat{\sigma} = (c_0, d_0) \cdots (c_{n-1}, d_{n-1})$ is an element in Σ^* where $|\hat{\sigma}| = n$ denotes the *length* of the sequence. An *infinite* sequence $\sigma = (c_0, d_0) \cdot (c_1, d_1) \cdots$ is an element of Σ^ω. We denote by $\sigma^i = (c_i, d_i)$ the $(i+1)^{th}$ letter in a (finite or infinite) sequence σ. Given an infinite sequence $\sigma \in \Sigma^\omega$, we denote by $\sigma^{[0,m)}$ the prefix of σ of length m, and by $\sigma^{m \cdots} = \sigma^m \sigma^{m+1} \ldots$ its infinite suffix from position m on. We denote by $\pi_i(\sigma) = i_0 \cdot i_1 \cdots$ the projection of sequence σ to its i^{th} component, where $i \in \{c, d\}$. We similarly define 1-dimensional sequences $\gamma \in \Gamma^\omega$ and $\hat{\gamma} \in \Gamma^*$. Given the alphabets Σ and Γ, a *transducer* is a function $F : \Sigma^\omega \to \Gamma^\omega$ that maps infinite sequences over the alphabet Σ into infinite sequences over Γ. In this paper, we restrict ourselves to sequential finite-state transducers. Such transducers can be seen as finite-state automata [Mea55] whose transitions are labeled by an input and an output letter (see Section IV).

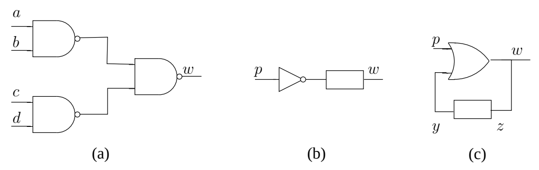


Figure 2. Circuits: (a) combinatorial (b) acyclic with delay (c) sequential

III. Sequential Circuits and their Robustness

We use sequential circuits [BW53] for modelling deterministic finite-state transducers with finite alphabet. We define robustness of sequential circuits with respect to the *common suffix distance* and study necessary and sufficient conditions for a circuit to be robust.

A. Sequential Circuits

A *combinatorial circuit* is a logic circuit that computes a Boolean function of its inputs. It consists of a set of *gates* and *inputs* interconnected by a set of *wires* without cycle. The gates are basic elements that compute simple Boolean functions such as NOT, AND or OR gates. Combinatorial circuits are *memoryless* by definition, meaning that the value of the output at any (discrete) time instant is a function of the values of its inputs at the same time instant. An example of combinatorial circuit with three NAND gates is shown in Figure 2(a).

A *sequential circuit* is an extension of combinatorial circuits with additional *memory devices*, called *delays*. The delay element "shifts" the input values by one time step, thus the output value of a memory device at time $t > 0$ is equal to its input value at time $t - 1$. If the circuit contains no cycle (as in Figure 2(b)), we call it an *acyclic circuit with delay*. In general, sequential circuits can contain cycles, as long as each cycle has at least one delay element. Cycles in sequential circuits are called *feedback loops* (as in Figure 2(c)). Feedback loops in sequential circuits are used to compute the value of the output at time $t > 0$ as a function of the current value of the inputs, but also of the value of its output at the previous time step $t - 1$ which is fed back to the circuit through the cycle. It follows that the output of a sequential circuit is a function of its *current* and *past* inputs, and sequential circuits are best viewed as mappings of *input sequences* into *output sequences*. The set of output values of the delay elements represent the current *state* of the circuit. A sequential circuit with k memory devices has at most 2^k states, or alternatively, a circuit with m states needs at least $\lceil \log m \rceil$ delay elements.

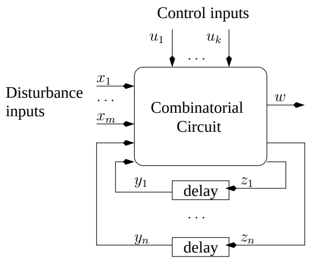


Figure 3. A generic sequential circuit $\mathcal{C}$.

We now formally define a sequential circuit as a system of equations that describe the relation between inputs, outputs and memory elements, using a standard notation [LMK98], [Brz62], with an additional distinction between *control* and *disturbance* input variables. Figure 3 shows a generic sequential circuit.

Definition 1: A *sequential circuit* $\mathcal{C}$ with $k+m$ control and disturbance inputs and n delay elements consists of sets $U = \{u_1, \ldots, u_k\}$ and $X = \{x_1, \ldots, x_m\}$ of *control* and *disturbance* variables, a set $Y = \{y_1, \ldots, y_n\}$ of *current-state* variables, a set $Z = \{z_1, \ldots, z_n\}$ of *next-state* variables, an *output* variable w, and a relation between input, output, and state variables expressed by a set of equations of the form

$$\begin{cases} z_1 &= f_1(u_1, \ldots, u_k, x_1, \ldots, x_m, y_1, \ldots, y_n) \\ & \ldots \\ z_n &= f_n(u_1, \ldots, u_k, x_1, \ldots, x_m, y_1, \ldots, y_n) \\ w &= f_\mathcal{C}(u_1, \ldots, u_k, x_1, \ldots, x_m, y_1, \ldots, y_n) \end{cases}$$

where f_i and $f_\mathcal{C}$ are Boolean functions, for $i = 1 \ldots n$. The set $\{f_1, \ldots, f_n\}$ is called the *transition equations* of $\mathcal{C}$ and $f_\mathcal{C}$ the *output equation* of $\mathcal{C}$. The next-state variables are updated according to the following equations, where y_i^t and z_i^t denote the valuation of variables y_i and z_i at time step t:

$$y_i^t = \begin{cases} 0 & \text{if } t = 0 \\ z_i^{t-1} & \text{if } t > 0 \end{cases} \quad \text{for } i = 1 \ldots n$$

The next lemma states that a sequential circuit can be "unfolded" at any time instant to express the output value as a function of its past and current (control and data) inputs, without explicit reference to the state variables.

Lemma 1: Given a sequential circuit $\mathcal{C}$ with $k+m$ control and disturbance input variables and n delay elements, and a time instant $t \in \mathbb{N}$, the value of the output and state variables in $\mathcal{C}$ at time t is a function of its past and current control and disturbance values, that is, there exist functions $f^t, g_i^t : \{0,1\}^{(k+m)\times(t+1)} \to \{0,1\}$ for $i = 1 \ldots n$ such that

$$w^t = f^t(u_1^0, \ldots, u_k^0, x_1^0, \ldots, x_m^0, \ldots, u_1^t, \ldots, u_k^t, x_1^t, \ldots, x_m^t)$$

and

$$z_i^t = g_i^t(u_1^0, \ldots, u_k^0, x_1^0, \ldots, x_m^0, \ldots, u_1^t, \ldots, u_k^t, x_1^t, \ldots, x_m^t)$$

Sequential circuits can also be encoded as functions that map input sequences to output sequences (i.e., transducers). Consider a sequential circuit $\mathcal{C}$ with $k+m$ control and disturbance input variables and n memory elements. Let $\Sigma_C = \{0,1\}^k$ and $\Sigma_D = \{0,1\}^m$ be the corresponding *control* and *disturbance alphabets*. We denote by $\Sigma = \Sigma_C \times \Sigma_D$ the joint *input* alphabet where each letter $(c,d) \in \Sigma$ denotes a vector of assignments to the input variables, and by $\Gamma = \{0,1\}$ the *output alphabet*. The sequential behavior of the circuit $\mathcal{C}$ is the function $F_\mathcal{C} : \Sigma^\omega \to \Gamma^\omega$ and we denote by $\gamma = F_\mathcal{C}(\sigma)$ the fact that the output sequence $\gamma \in \Gamma^\omega$ is generated by $\mathcal{C}$ on input $\sigma \in \Sigma^\omega$. By Lemma 1, for all $t \geq 0$, we can express γ^t as a function of the previously consumed input letters $\sigma^{[0,t+1)}$. In the rest of the paper, we use this sequence-oriented definition of the semantics of sequential circuits.

B. Hamming Distance

Hamming distance is a standard metric that has been proposed to measure the similarities between pairs of sequences.

Definition 2: Let Σ be a finite alphabet and $a_1, a_2 \in \Sigma$. The *Hamming distance* between two finite words $\hat{\sigma}_1, \hat{\sigma}_2 \in \Sigma^*$ such that $|\hat{\sigma}_1| = |\hat{\sigma}_2|$, is defined inductively by

$$\begin{aligned} d_H(\epsilon, \epsilon) &= 0 \\ d_H(a_1 \cdot \hat{\sigma}_1, a_2 \cdot \hat{\sigma}_2) &= \begin{cases} d_H(\hat{\sigma}_1, \hat{\sigma}_2) & \text{if } a_1 = a_2 \\ 1 + d_H(\hat{\sigma}_1, \hat{\sigma}_2) & \text{if } a_1 \neq a_2 \end{cases} \end{aligned}$$

The Hamming distance between two infinite words $\sigma_1, \sigma_2 \in \Sigma^\omega$ is

$$d_H^\omega(\sigma_1, \sigma_2) = \lim_{n \to \infty} d_H(\sigma_1^{[0,n)}, \sigma_2^{[0,n)}).$$

We argue that the Hamming distance may not always provide a satisfactory notion of robustness for sequential circuits. We illustrate this point with the sequential circuit $\mathcal{C}$ shown in Figure 4. This circuit has a single disturbance variable p and an output variable w and behaves as follows: (1) $\mathcal{C}$ outputs 1 whenever p is *true*; (2) the first time that the input p becomes *false*, $\mathcal{C}$ outputs either 0 or 1, depending on whether the current value of the input sequence p was preceded by an even or odd number of consecutive 1s (3) the output will be 1 no matter the subsequent inputs. Consider the following patterns of input sequences σ_1 and σ_2 and the corresponding output sequences $\gamma_1 = f_\mathcal{C}(\sigma_1)$ and $\gamma_2 = f_\mathcal{C}(\sigma_2)$ generated by $\mathcal{C}$, where $n \geq 0$ is some arbitrary integer

$$\begin{array}{ll} \sigma_1 : \ \overline{\boldsymbol{p}} \cdot p^{2n} \cdot \overline{p} \cdot p^\omega & \gamma_1 : \ w \cdot w^{2n} \cdot \boldsymbol{w} \cdot w^\omega \\ \sigma_2 : \ \boldsymbol{p} \cdot p^{2n} \cdot \overline{p} \cdot p^\omega & \gamma_2 : \ w \cdot w^{2n} \cdot \overline{\boldsymbol{w}} \cdot w^\omega \end{array}$$

In the above example $d_H(\sigma_1, \sigma_2) = 1$ and $d_H(\gamma_1, \gamma_2) = 1$, for any value of n. It follows that for this particular pattern of inputs, the circuit $\mathcal{C}$ is "robust" with respect to the Hamming distance. However, the effect of the mismatch in the inputs becomes observable in the outputs only after $2n + 2$ steps, where n can be arbitrarily large. This unbounded "noise"

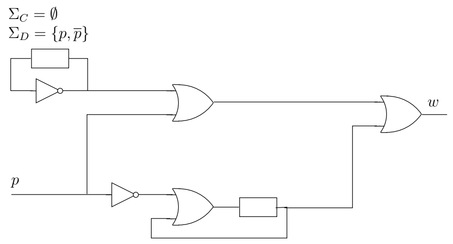


Figure 4. A sequential circuit $\mathcal{C}$ that delays the mismatching input letter in $\sigma_1 : \overline{p} \cdot p^{2n} \cdot p \cdot p^{\omega}$ and $\sigma_2 : p \cdot p^{2n} \cdot p \cdot p^{\omega}$ arbitrarily far in the output.

propagation may not be appropriate for analysis of critical systems where one would expect the system to recover from a noisy input in a predictable amount of time.

C. Common suffix distance

We propose an alternative metric that we call the *common suffix distance* and that is the last position in which two sequences differ. We note that the common suffix distance coincides with the inverse of the Cantor distance.

Definition 3: Let Σ be a finite alphabet. The *common suffix distance* between two finite words $\hat{\sigma}_1, \hat{\sigma}_2 \in \Sigma^*$ such that $|\hat{\sigma}_1| = |\hat{\sigma}_2|$ is defined inductively by

$$\begin{array}{lcl} d_s(\epsilon, \epsilon) & = & 0 \\ d_s(\hat{\sigma}_1 \cdot a_1, \hat{\sigma}_2 \cdot a_2) & = & \left\{ \begin{array}{ll} d_s(\hat{\sigma}_1, \hat{\sigma}_2) & \text{if } a_1 = a_2 \\ |\hat{\sigma}_1| + 1 & \text{if } a_1 \neq a_2 \end{array} \right. \end{array}$$

The common suffix distance between two infinite sequences $\sigma_1, \sigma_2 \in \Sigma^{\omega}$ is

$$d_s^{\omega}(\sigma_1, \sigma_2) = \lim_{n \to \infty} d_s(\sigma_1^{[0,n)}, \sigma_2^{[0,n)}).$$

The common suffix distance is an upper bound on the Hamming distance, as for all sequences $\sigma_1, \sigma_2 \in \Sigma^{\omega}$, $d_H(\sigma_1, \sigma_2) \leq d_s(\sigma_1, \sigma_2)$. Although the common suffix distance does not count the number of relative differences between σ_1 and σ_2 within their prefix where the mismatching occur, it does provide enough information for checking the robustness of a sequential circuit with respect to a subset of its input variables.

The following lemma states that the distance between two sequences $\sigma_1, \sigma_2 \in \Sigma^{\omega}$ is finite (and bounded by some integer k) if and only if σ_1 and σ_2 have common suffixes from a position strictly smaller than k.

Lemma 2: For all $\sigma_1, \sigma_2 \in \Sigma^{\omega}$ and $k > 0$, $d_s^{\omega}(\sigma_1, \sigma_2) < k$ iff there exists $0 \leq m < k$ such that $\sigma_1^{m\cdots} = \sigma_2^{m\cdots}$.

We can show that the common suffix distance defines a metric.

Lemma 3: The common suffix distance is a metric.

D. Σ_D-Robustness as Finite Disturbance Horizon

In this section, we define robustness for sequential circuits as a form of continuity with respect to the disturbance subset of its input variables in the common suffix topology. Our definition relates the positions of mismatching characters in the pairs of input and output sequences, where the two input sequences have the identical control, but possibly different disturbance components. In particular, we ask that there exists a bound b on the propagations of mismatches, namely, that if the last mismatching in the input sequence occurs at position k, then the last mismatching in the output sequence occurs before position $k + b$. The addition is used in order to make the definition of robustness invariant to the actual positions in which mismatches happen.

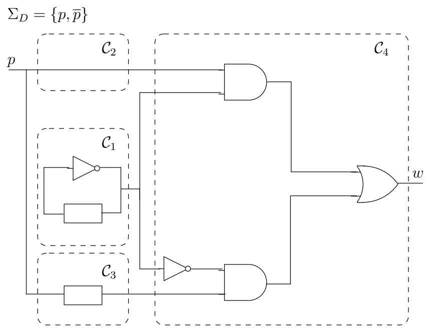


Figure 5. An example of a robust sequential circuit

Definition 4: A sequential circuit $\mathcal{C}$ with Σ_C and Σ_D control and disturbance input alphabets is Σ_D*-robust* if there exists $b \geq 0$ such that for all $k > 0$ and all $\sigma_1, \sigma_2 \in (\Sigma_C \times \Sigma_D)^{\omega}$ such that $\pi_C(\sigma_1) = \pi_C(\sigma_2)$, if $d_s^{\omega}(\sigma_1, \sigma_2) < k$, then $d_s^{\omega}(F_{\mathcal{C}}(\sigma_1), F_{\mathcal{C}}(\sigma_2)) < k + b$.

We introduce the class of sequential circuits with *finite disturbance horizon*. A circuit with control alphabet Σ_C and disturbance alphabet Σ_D has the finite disturbance horizon property if it has a fixed bound on the number of current and past disturbance inputs that are necessary to determine its output at any time instant.

Definition 5: Let $F_{\mathcal{C}} : (\Sigma_C \times \Sigma_D)^{\omega} \to \Gamma^{\omega}$ be the sequential function of a circuit $\mathcal{C}$, where $\gamma = F_{\mathcal{C}}(\sigma)$. We say that $\mathcal{C}$ has *finite disturbance horizon* if for some bound b, for all time instants t greater than or equal to b, it is sufficient to consider, together with all past control input symbols, the current and previous b disturbance input symbols in order to compute the value of γ^t, i.e., there exists $b \in \mathbb{N}$ such that for all $t \geq b$ and $\sigma \in (\Sigma_C \times \Sigma_D)^{\omega}$, there exists a function $G^t : \Sigma_C^{t+1} \times \Sigma_D^{b+1} \to \Gamma$, such that $\gamma^t = G^t(\sigma_C^{[0,t+1)}, \sigma_D^{[t-b,t+1)})$, where $\sigma_C = \pi_C(\sigma)$ and $\sigma_D = \pi_D(\sigma)$.

Note that the above definition considers time instants greater than or equal to b. For $t < b$, the number of previous (control and disturbance) inputs needed to compute the output is trivially bounded by b.

Example 1: The circuit shown in Figure 2 (c) contains a feedback loop and $u^t = \bigvee_{i=0}^{t} p^i$ for all $t \geq 0$, that is the output at time t depends on the value of *all* the previous values of p. This circuit is robust only if p is a control and not a disturbance variable.

Example 2: The circuit in Figure 5 has a disturbance variable p and a feedback loop and can be logically decomposed into four components. The circuit $\mathcal{C}_1$ implements a modulo 2 counter using a feedback loop and has no input. The circuits $\mathcal{C}_2$ and $\mathcal{C}_3$ define two sequential functions that are dependent only on the input value in the present and previous time step, respectively. Finally, $\mathcal{C}_4$ propagates the output of either $\mathcal{C}_2$ or $\mathcal{C}_3$ depending on the current value of the counter $\mathcal{C}_1$. The output can be expressed as $w^t = p^t$ if t is even, and $w^t = p^{t-1}$ if t is odd. Therefore, the whole system implements a sample-and-hold function that propagates the input value at every even time step, and holds it for two instants of time. Although the output at time t depends on the current state of the circuit, it can be computed as a function of some bounded portion of the past inputs. Therefore the circuit has finite disturbance memory.

We show in Theorem 1 that a sequential circuit is Σ_D-robust if and only if it has finite disturbance horizon, that is if it always forgets about disturbance inputs older than some bounded amount of time. The intuition is that an "altered" disturbance input is forgotten by the circuit after some finite time and consequently does not influence the computation of the circuit's output after that time.

Theorem 1: A sequential circuit is Σ_D-robust if and only if it has finite disturbance horizon.

The next result follows from the fact that combinatorial circuits are memoryless and that acyclic circuits with delay have finite disturbance horizon by definition.

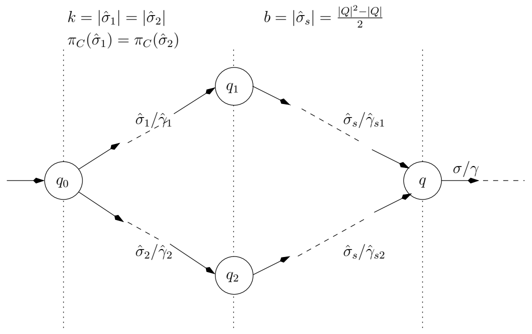
Corollary 1: Every combinatorial circuit and every acyclic circuit with delay is Σ_D-robust.

IV. Operational Characterization of Σ_D-Robustness for Sequential Circuits

Sequential circuits have an equivalent graphical representation called *Mealy machines* [Mea55]. Such machines consist of a finite number of states (the internal memory of the circuit) and transitions between the states labeled by input/output actions. Mealy machines can be used to model the high-level behavior of the underlying circuit. We propose an alternative characterization of robustness based on the state-based properties of Mealy machines.

Definition 6: A *Mealy machine* is a tuple $\mathcal{M} = (Q, \Sigma_C, \Sigma_D, \Gamma, q_0, \delta, \lambda)$ where Q is a finite set of states, Σ_C and Σ_D are control and disturbance input alphabets, Γ is the output alphabet, $q_0 \in Q$ is the initial state, $\delta : Q \times (\Sigma_C \times \Sigma_D) \rightarrow Q$ is the state transition function and $\lambda : Q \times (\Sigma_C \times \Sigma_D) \rightarrow \Gamma$ is the output function.

The state transition function δ and the output function λ are extended to sequences as follows (inductively): $\delta(q, (c,d) \cdot \hat{\sigma}) = \delta(\delta(q,(c,d)), \hat{\sigma})$ and $\lambda(q, \hat{\sigma} \cdot (c,d)) = \lambda(\delta(q,\hat{\sigma}), (c,d))$ for all $(c,d) \in (\Sigma_C \times \Sigma_D)$ and $\hat{\sigma} \in (\Sigma_C \times \Sigma_D)^+$.



$k = |\hat{\sigma}_1| = |\hat{\sigma}_2|$
$\pi_C(\hat{\sigma}_1) = \pi_C(\hat{\sigma}_2)$
$b = |\hat{\sigma}_s| = \frac{|Q|^2 - |Q|}{2}$
q_0
q_1
q_2
q
$\hat{\sigma}_1/\hat{\gamma}_1$
$\hat{\sigma}_s/\hat{\gamma}_{s1}$
$\hat{\sigma}_2/\hat{\gamma}_2$
$\hat{\sigma}_s/\hat{\gamma}_{s2}$
σ/γ

Figure 6. Σ_D-synchronization of Mealy machines: General case

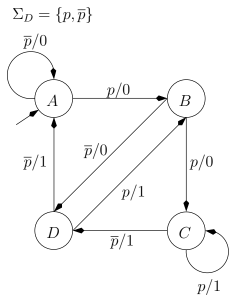


Figure 7. Σ_D-synchronization of Mealy machines: Example

A Mealy machine $\mathcal{M}$ defines a function $f_{\mathcal{M}} : (\Sigma_C \times \Sigma_D)^\omega \rightarrow \Gamma^\omega$ in the expected way. The machine $\mathcal{M}$ and a sequential circuit $\mathcal{C}$ are *equivalent* if $f_{\mathcal{M}}(\sigma) = F_{\mathcal{C}}(\sigma)$ for all $\sigma \in (\Sigma_C \times \Sigma_D)^\omega$. $\mathcal{M}$ is *minimal* if for all states $q, q' \in Q$ such that $q \neq q'$, there exists an input sequence $\sigma_a \in (\Sigma_C \times \Sigma_D)^+$ such that $\lambda(q, \sigma_a) \neq \lambda(q', \sigma_a)$. Mealy machines can be minimized in time $O(n \cdot \log n)$ where $n = |Q|$ is the number of states [Hop71].

In the sequel, we define a class of Mealy machines (as in Figure 6) that we call Σ_D*-synchronized*. Intuitively, a Mealy machines is Σ_D-synchronized if from every pair of states reachable by two input sequences of the same length and with the same control component, a "reset" state is reached after reading any sufficiently long word. Consider the machine $\mathcal{M}$ in Figure 7 with $\Sigma_D = \{p, \overline{p}\}$, implementing a delay of length 2. From the initial state A, every state is reachable with some input of length 3. In this particular example, in order to show that the machine is Σ_D-synchronized, we need to check that from every pair of its states, a single state is reached with every sufficiently long word. This holds since from every state, the disturbance input $\overline{p} \cdot \overline{p}$ leads to A, $\overline{p} \cdot p$ to B, $p \cdot \overline{p}$ to C, and $p \cdot p$ to D.

Definition 7: A Mealy machine $\mathcal{M}$ is Σ_D*-synchronized* if there exists a bound $\beta \in \mathbb{N}$ such that for all $\hat{\sigma}_1, \hat{\sigma}_2, \hat{\sigma}_s \in (\Sigma_C \times \Sigma_D)^*$ with $|\hat{\sigma}_1| = |\hat{\sigma}_2|$, $\pi_C(\hat{\sigma}_1) = \pi_C(\hat{\sigma}_2)$ and $|\hat{\sigma}_s| \geq \beta$, we have $\delta(q_0, \hat{\sigma}_1 \cdot \hat{\sigma}_s) = \delta(q_0, \hat{\sigma}_2 \cdot \hat{\sigma}_s)$.

It turns out that the class of Σ_D-synchronized Mealy machines corresponds exactly to the one of sequential circuits that are Σ_D-robust.

Theorem 2: A sequential circuit is Σ_D-robust if and only if its equivalent Mealy machine is Σ_D-synchronized.

Moreover, the robustness bound b in Definition 4 and the bound β in Definition 7 coincide. Finally, we show in the proof of Theorem 2 in Appendix **??** that if the Mealy machine of a sequential circuit is Σ_D-synchronized, then we can take $\beta = \frac{|Q|^2-|Q|}{2}$ giving the same value for the robustness bound b of the circuit.

V. Checking Σ_D-Robustness of Mealy Machines

In this section, we propose a two-step algorithm for checking the Σ_D-robustness of minimal Mealy machines. The algorithm is based on a standard graph analysis. Given an arbitrary minimal Mealy machine $\mathcal{M} = (Q, \Sigma_c, \Sigma_d, \Gamma, q_0, \delta, \lambda)$, the first part of the algorithm is a graph exploration procedure that computes $Q_{\mathcal{M}}$, the set that contains all distinct pairs of states in $\mathcal{M}$ that are reachable from the initial state q_0 by reading two input words of the same length with the same control, but possibly differing disturbance components.

$$\begin{aligned} Q_{\mathcal{M}} \quad = \quad & \{\{q_1, q_2\} \mid q_1, q_2 \in Q \text{ s.t. } q_1 \neq q_2 \text{ and} \\ & \exists \sigma_a, \sigma_b \in (\Sigma_C \times \Sigma_D)^* \text{ s.t.} |\sigma_a| = |\sigma_b|, \\ & \pi_C(\sigma_a) = \pi_C(\sigma_b) \\ & \text{and } q_1 = \delta(q_0, \sigma_a) \text{ and } q_2 = \delta(q_0, \sigma_b)\} \end{aligned}$$

We have to check whether there exists a length β such that on *all* input words $\sigma_s \in (\Sigma_C \times \Sigma_D)^*$ of length β, from every pair of states $\{q_1, q_2\}$ in $Q_{\mathcal{M}}$, a single state is reached, i.e., $\delta(q_1, \sigma_s) = \delta(q_2, \sigma_s)$. This is the case only if there exists no word σ_s of length $\frac{|Q|^2-|Q|}{2}$ that induces a loop through pairs in $Q_{\mathcal{M}}$ (otherwise an arbitrarily long word can be constructed that violates the condition of Σ_D-synchronization), i.e., that $\forall \{q_1, q_2\} \in Q_{\mathcal{M}} \cdot \nexists\ \sigma_s : q_1 = \delta(q_1, \sigma_s)$ and $q_2 = \delta(q_2, \sigma_s)$. It follows that the Σ_D-robustness of a Mealy machine $\mathcal{M}$ can be decided by a cycle detection algorithm on the graph $G_{\mathcal{M}}(Q_{\mathcal{M}}, \delta_{\mathcal{M}})$ where $\delta_{\mathcal{M}} \subseteq Q_{\mathcal{M}} \times Q_{\mathcal{M}}$ and $(\{q_1, q_2\}, \{q_1', q_2'\}) \in \delta_{\mathcal{M}}$ iff $q_1' = \delta(q_1, (c, d))$ and $q_2' = \delta(q_2, (c, d))$ and $q_1' \neq q_2'$ for some $(c, d) \in \Sigma_C \times \Sigma_D$. The cycle detection can be done on-the-fly with the generation of $G_{\mathcal{M}}$.

Theorem 3: The complexity of checking robustness of a minimal Mealy machine $\mathcal{M} = (Q, \Sigma_C, \Sigma_D, \Gamma, q_0, \delta, \lambda)$ is $\mathcal{O}(\frac{|Q|^2+|Q|}{2} \cdot |\Sigma_C| \cdot |\Sigma_D|^2)$.

VI. Detailed Example

An *adder-subtractor* is a sequential circuit that combines addition or subtraction operations of binary numbers in a single unit. Adders and subtractors are usually part of the *arithmetic logic unit* (ALU). There is an additional *control unit* that decides which operation the ALU should perform.

The 4-bit adder-subtractor shown in Figure VI, the circuit is composed of four 1-bit full adders and a control unit *switch*. It combines the adding and subtracting capabilities by exploiting the fact that when the numbers are in two's complement, it is sufficient to invert each bit with a NOT gate and add 1, to implement subtraction using the adders. The circuit takes as inputs variables $a_0, \dots, a_3$ and $b_0, \dots, b_3$ and outputs their sum or difference in the output variables $s_0, \dots, s_3$, together with a carry output c_4.

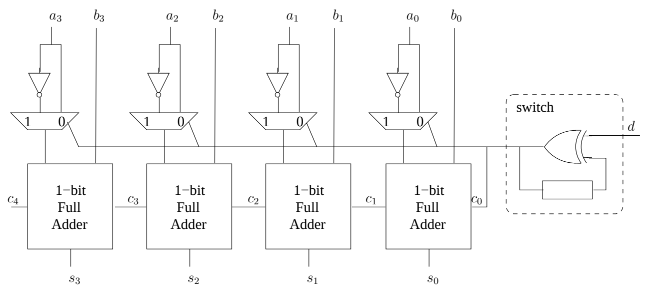


Figure 8. A 4-bit ripple carry adder-subtractor

The function that is computed by the circuit depends on the *switch* control unit that takes as input the signal d. Initially, the circuit is set to do addition. Whenever d is set to high, the circuit switches to the other function.

Intuitively, the input variables in this circuit are naturally partitioned into sets $U = \{d\}$ and $X = \{a_0, b_0, \dots, a_3, b_3\}$ of control and disturbance variables, respectively. The circuit is clearly Σ_D-robust under this partition, because 1-bit full adders are combinatorial circuits without memory. However, if we assume that the external environment is not fully reliable, and may introduce errors in the d signal, we cannot consider d anymore a control, but rather a disturbance variable. In that case, the circuit is not Σ_D-robust anymore. For this, consider two input sequences of d, $\sigma_d : 110^\omega$ and $\sigma_d' : 10^\omega$. In the first case, the circuit switches to do the subtraction in the first step, and than switches to the addition operation in the next step and remains doing addition forever. On the other hand, the second input sequence makes the circuit doing subtraction forever. As a consequence, the common suffix distance between the two input sequences is 1, but the output sequences drastically differ, given that they are computed using a different operation.

This circuit can be made Σ_D-robust with respect to all inputs by changing the encoding of the control signal d. Instead of having the environment that provides the information when to switch from addition to subtraction and vice versa, we can design a circuit that expects from the environment explicit request about which operation to apply at each step. In this case, we do not need the switch control unit and the control input can be directly fed to the ALU. Any error in the control signal would then affect only the computation of the output at the time step of the error occurrence, but would not propagate to the future computation of the output.

VII. Future Work

A natural extension of our work would be to study the logical and algebraic characterization of robustness and identify the subset of ω-regular languages that are robust with respect to the distance measures described in this paper. Another important direction for future research is to study quantitative extensions of sequential circuits such as continuous-time sequential circuits [PRT04] or probabilistic circuits,[PM75]. One could obtain a logical characterization of such circuits by using the quantitative and probabilistic languages introduced in [CDH08], [CDH09].

We would also like to extend the study of robustness to variants of asynchronous circuits. We are particularly interested in considering *latency-insensitive designs* [CMSV01] and *elastic circuits* [KCKO06], which relax the standard synchronous model by elasticizing the time dimension and enabling the circuit to be tolerant to arbitrary discrete latencies in its timing behavior. In order to study such circuits, we believe that we should adapt our notion of robustness to the *edit*-like distances that allow not only standard substitution, but also insertion and deletion of data.

It would be also interesting to be able to distinguish between systems that are robust. As an example, consider two robust circuits $\mathcal{C}_1$ and $\mathcal{C}_2$. Assume that the two circuits generate the same behaviors for a given subset I of expected inputs, but may generate different behaviors for inputs that are outside of I. It could be that the b in the distance is the same for both $\mathcal{C}_1$ and $\mathcal{C}_2$, but that this b is only reached when reading a few sequences outside of I in $\mathcal{C}_1$ and all the sequences outside of I in $\mathcal{C}_2$. In this situation, one should conclude that it is better to use $\mathcal{C}_1$ than $\mathcal{C}_2$. Indeed, for input sequences that are outside of I, $\mathcal{C}_1$ recovers faster than $\mathcal{C}_2$. The common suffix distance cannot make such distinctions and should be further refined in order to be able to quantify the degree of robustness of such circuits.

References

[BGHJ09] R. Bloem, K. Greimel, T. A. Henzinger and B. Jobstmann, Synthesizing Robust Systems, In *FMCAD'09*, 2009.

[Brz62] J. A. Brzozowski, *Regular Expression Techniques for Sequential Circuits*, PhD Dissertation, University of Priceton, 1962.

[BW53] A. W. Burks and J. .B. Wright, Theory of logical nets, In *IRE'53*, 1357–1365, 1953.

[CB02] P. Caspi and A. Benveniste, Toward an approximation theory for computerised control, In *EMSOFT02*, 2491, 2002.

[CDH08] K. Chatterjee, L. Doyen and T. A. Henzinger, Quantitative Languages, IN *CSL'08*, 385–400.

[CDH09] K. Chatterjee, L. Doyen and T. A. Henzinger, Probabilistic Weighted Automata, In *CONCUR'09*, 244–258, 2009.

[CMSV01] L. P. Carloni, K. L. McMillan and A. L. Sangiovanni-Vincentelli, Theory of latency-insensitive design, In *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 20(9):1059-1076, 2001.

[DM94] B. Delyon and O. Maler, On the Effects of Noise and Speed on Computations, In *Theoretical Computer Science*, 129, 279-291, 1994

[GHJ97] V. Gupta, T. A. Henzinger and R. Jagadeesan, Robust timed automata. In *Hybrid and Real-Time Systems*, LNCS 1201, 331–345, 1997.

[Ham50] R. W. Hamming, Error detecting and error correcting codes. In *Bell System Technical Journal*, 29 (2), 147160, 1950.

[Hen08] T. A. Henzinger, Two challenges in embedded systems design: Predictability and robustness, In *Philosophical Transactions of the Royal Society*, 366, 3727–3736, 2008.

[Hop71] J. E. Hopcroft, An $n \cdot \log n$ algorithm for minimizing states in a finite automaton, *Technical Report CS-71-190*, Stanford University, 1971.

[KC04] C. Kossentini and P. Caspi, Approximation, sampling and voting in hybrid computing systems, In *HSCC'04*, 363–376, 2004.

[KCKO06] S. Krstic, J. Cortadella, M. Kishinevsky and J. O'Leary, Synchronous elastic networks, In *FMCAD'06*, 19–30, 2006.

[KLS08] O.Kupferman, W.Li and S.A.Seshia, A theory of mutations with applications to vacuity, coverage, and fault tolerance, In FMCAD '08, 1–9, 2008.

[Lev66] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10, 707710, 1966.

[Lev96] W. S. Levine, *The Control Handbook*. New York: CRC Press, 1996.

[LMK98] G. Langholz, J. L. Mott and A. Kandel, *Foundations of Digital Logic Design*, World Scientific Publishing, 1998.

[Mal93] S. Malik, Analysis of cyclic combinatorial circuits, In *ICCAD'93*, 618–625, 1993.

[Mea55] G. H. Mealy, A method for synthesizing sequential circuits, In *Bell Systems Technical Journal*, 1045–1079, 1955.

[PM75] K. P. Parker and E. J. McCluskey, Analysis of Logic Circuits with Faults Using Input Signal Probabilisties, In *IEEE transaction in Computer Science*, 573–578, C-24, 1975.

[PRT04] D. Pardo, A. M. Rabinovich and B. A. Trakhtenbrot, Synchronous circuits over continuous time: feedback reliability and completeness, In *Fundamenta Informaticae*, 62(1), 123–137, 2004.

[SLM09] S. A. Seshia, W. Li and S. Mitra, Verification-guided soft error resilience, In *DATE '07*, 1442–1447, 2007.

[Vol08] M. V. Volkov, Synchronizing automata and the Černỳ conjecture. In *LATA'08*, 11–27, 2008.