TIMED LOGIC CONFORMANCE AND ITS APPLICATION

Frank C. D. Young[1] *Kenneth S. Stevens*[2] *Robert P. Graham Jr.*[3]

[1]Air Force Research Laboratory, Wright-Patterson AFB, OH 45433, USA
[2]SCL, Intel Corporation, Hillsboro OR, 97124, USA
[3]Air Force Institute of Technology, Wright-Patterson AFB, OH 45433, USA

ABSTRACT

Timed Logic Conformance (TLC) is a bisimulation-style partial order relationship defined over the statespace of Timed Safety Automata (TSA) with real-valued clocks. In contrast to timed simulation, Calculus of Timed Refinement (CTR), and Time-Abstracted bisimulation, TLC defines when one system is an acceptable implementation of another by asymmetric bisimulation-style requirements for specification inputs and implementation outputs. While TLC does not necessarily preserve timed properties, it intuitively and pragmatically supports writing abstract specifications and verifying them against implementations. TLC scales up by substituting verified specifications for implementations and hierarchically verifying larger systems. TLC verification is an alternative to assumes-guarantees reasoning process. TLC verification depends on explicitly capturing environmental timing properties in the specification and insuring they are satisfied in the TLC relation. The region-automata-based TLC System (TLCS) implements TSA parallel composition and a TLC decision procedure which is used to hierarchically verify the STARI queue.

1. INTRODUCTION

In contrast to timed simulation [TAKB96], Calculus of Timed Refinement (CTR) [C̈95], and Time-Abstracted bisimulation [LY93] Timed Logic Conformance (TLC) defines when one system is an acceptable implementation of another by associating asymmetric simulation requirements for matching specification (spec) inputs and implementation (imp) outputs. In general, the imp must match all spec inputs and outputs, and the spec must allow all reachable outputs that the imp produces. Extra imp input derivatives do not matter, and reachable imp outputs must be a timed subset of spec outputs (i.e. no imp outputs may occur outside the temporal bounds of the same output in the spec). Until it must output, the imp must accept all inputs that the spec accepts, so spec inputs must be a timed subset of imp inputs. TLC does not necessarily preserve timed properties because it accepts imps that may accept more inputs than the spec. TLC is weaker than traditional weak timed bisimulation but not directly comparable to most other timed equivalence relations because of TLC's asymmetry. TLC and parallel composition hierarchically break verification down into independent TLC relations between subcomponent specs and imps to lower levels of abstraction.

2. TIMED SAFETY AUTOMATA

In this research, we use a flavor of TSA with both location and transition predicates and action-labeled transitions. For a formal exposition of TSA expressiveness and computational complexity see [AD94]. Our TSA definition is based on Sokolsky's [SS95], and supports a dense-time model of time with the non-negative real numbers $\mathbb{R} \triangleq [0, \infty)$, and time constants from the non-negative integers $\mathcal{Z} \triangleq \{0, 1, 2, ...\}$.

2.1. Basic TSA Definitions

Let $\mathcal{T}$ denote the set of all TSA automata. Given: **Clock:** a $\mathbb{R}$-valued variable, and let $\mathcal{C}$ be the set of all clock variables. **Clock constraint:** C is an expression of the form $x\ R\ c$ where $x \in \mathcal{C}$ and $c \in \mathcal{Z}$ and $R \in \{\leq, \geq, <, >\}$. **Clock assignment:** $\vec{\pi} = (x_1, ..., x_n)$ an n-dimensional point in $\mathbb{R}^n$, where each real number x_i is the $\mathbb{R}$-value mapped from clock c_i in the n-sized set of clocks used to constrain the TSA. **Idling:** $\vec{\pi} + d = (x_1 + d, ..., x_n + d)\ d \in \mathbb{R}$. **Clock reset:** $\vec{\pi}[\eta := 0].\ \eta \subseteq \mathcal{C}$ where $\forall\ x_i \in \vec{\pi}[c_i \in \eta \Rightarrow x_i = 0]$; otherwise x_i is the same before and after the reset. **Region:** $\rho \subseteq \mathbb{R}^n$ formed by a conjunction of clock constraints. Let Ξ be the set of all regions. **Input action—name:** $a \in \mathcal{A}$. **Output action—coname:** $\overline{a} \in \overline{\mathcal{A}}, \overline{\overline{a}} = a$. **Labels:** $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$. **Invisible internal action:** $\tau \notin \mathcal{L}$. **Location:** $\langle l, \rho_l \rangle$, where l is unique location name, and ρ_l is a past-closed region called a **location invariant**. A region ρ is **past-closed** when it includes time $\vec{0}$ i.e.

$$\forall\ \vec{p} \in \rho[\forall\ \vec{d} \in \mathbb{R}^n[\vec{0} \leq \vec{d} \leq \vec{p} \Rightarrow \vec{d} \in \rho]]$$

A TSA T is a 5-tuple $\langle \Lambda, Act, \xi, \langle l_0, \rho_0 \rangle, \longmapsto \rangle$, **1.** Λ a finite set of locations. **2.** $Act = \mathcal{L} \cup \{\tau\}$, a set of actions ranged over by α. **3.** $\xi \subseteq \mathcal{C}$, a size-$n$ set of $\mathbb{R}$-value clocks. **4.** $\langle l_0, \rho_0 \rangle \in \Lambda$, the start location, where initially $\vec{\pi} \in \rho_0 = \vec{0}$. **5.** $\longmapsto \subseteq \Lambda \times Act \times \Xi \times \wp(\xi) \times \Lambda$, the transition relation, where each transition is labeled by an action, a region (called a **guard**), and a set of clocks which are reset to 0 when the transition occurs.

Guards are interpreted as necessary conditions for the transition to occur, and invariants are interpreted as sufficient conditions for a transition to occur [HNSY94]; i.e. time passing forces a change of location to avoid $\vec{\pi} \notin \rho_l$. Invariants are also necessary conditions for the TSA to be in the associated location. Unspecified guards and invariants are defined to be always satisfied. Informally, TSA take

instantaneous transitions from location to location. When no transitions occur, TSA idle in a location $\langle l, \rho_l \rangle$ passing time by incrementing all clocks $x_i \in \vec{\pi}$ by $d \in I\!R$ such that l's location invariant is satisfied—i.e. $\vec{\pi} + d \in \rho_l$. Without loss of generality, we consider only non-Zeno TSA. Non-Zenoness is a liveness condition that asserts time can always progress [HNSY94].

Figure 1 is an inverter TSA with delay bounds from `MinD` to `MaxD`, clock `k`, input `a`, output `b_`, (underscores denote output labels) and four locations labeled with the value of the input and output signals. **Unstable locations**[1] `00` and `11` have invariant `k`≤`MaxD`, outputs from those locations have guard `k`≥`MinD`, the inverter initializes in stable locations `01` or `10`, and that stable-location `a` input transitions reset clock `k`.

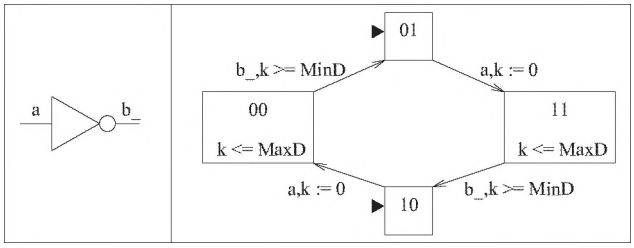

Figure 1. Simple Inverter Logic Symbol and TSA

Specifying the behavior of complex imps as single flat TSA is too tedious. Precise mathematical definitions for hierarchical parallel composition based on DLTS semantics and EBNF productions defining the syntax of TLCS parallel TSA are included in [Youss].

2.2. TSA Semantics

We define the TSA semantics via a Dense Labeled Transition System (DLTS) automata with uncountable state sets.

Let $\mathcal{D}$ denote the set of all DLTS automata. Every TSA $T = \langle \Lambda, Act, \xi, \langle l_0, \rho_0 \rangle, \longmapsto \rangle$ induces a DLTS automaton $D = \langle S, Act, \longrightarrow, \langle l_0, \vec{0} \rangle \rangle$ such that:

1. S is a set of timed states defined by the following rule:

$$\forall \langle l, \rho_l \rangle \in \Lambda [\vec{\pi} \in \rho_l \Rightarrow \langle l, \vec{\pi} \rangle \in S] \quad (1)$$

We sometimes use S_I and S_S to distinguish between the imp and spec DLTS state-spaces. **2.** $Act = \mathcal{L} \cup \{\tau\}$ a set of actions ranged over by α. **3.** $\langle l_0, \vec{0} \rangle$ the start state assigning 0 to every clock. **4.** $\longrightarrow \subseteq S \times (Act \cup I\!R) \times S$ is a transition relation defined by the following two rules for every $\langle l, \rho_l \rangle \in S$:

$$\langle l, \rho_l \rangle \overset{\alpha, \rho, \eta}{\longmapsto} \langle l', \rho_{l'} \rangle \wedge \vec{\pi} \in \rho \wedge \vec{\pi} \in \rho_l \wedge \vec{\pi}[\eta := 0] \in \rho_{l'} \Rightarrow \langle l, \vec{\pi} \rangle \overset{\alpha}{\longrightarrow} \langle l', \vec{\pi}[\eta := 0] \rangle \quad (2)$$

$$\delta \in I\!R \wedge \vec{\pi}, \vec{\pi} + \delta \in \rho_l \Rightarrow \langle l, \vec{\pi} \rangle \overset{\delta}{\longrightarrow} \langle l, \vec{\pi} + \delta \rangle \quad (3)$$

In Rule 2, D transitions from location l to l' via action α. No time passes, but all clocks in $\eta \subseteq \xi$ are reset to 0. Clock assignment $\vec{\pi}$ must satisfy (be an element of) regions ρ_l and ρ, and clock reset $\vec{\pi}[\eta := 0]$ must satisfy $\rho_{l'}$. Under rule 2, we call timed state $\langle l', \vec{\pi}[\eta := 0] \rangle$ a **transition successor** of timed state $\langle l, \vec{\pi} \rangle$.

In Rule 3, D stays in location l with time delay δ if both $\vec{\pi}$ and $\vec{\pi} + \delta$ satisfy ρ_l. Under rule 3, we call timed state $\langle l', \vec{\pi} + \delta \rangle$ a **time successor** of timed state $\langle l, \vec{\pi} \rangle$.

Possible Function (Ψ): The overloaded function Ψ computes sets of possible actions from TSA locations and DLTS timed states as follows (underscores in tuples are don't-cares): **1.** Let $\Psi : \Lambda \rightarrow \wp(Act)$ return the set of actions possible from a TSA location: $\Psi(l) \triangleq \{\sigma \mid \langle l, \sigma, \_, \_, \_ \rangle \in \longmapsto\}$ **2.** Let $\Psi : S \rightarrow \wp(Act \cup I\!R)$ return the set of actions possible from a DLTS timed state such that $\Psi(s) \triangleq \{\sigma \mid \langle s, \sigma, \_ \rangle \in \longrightarrow\}$

2.3. Region Automata

Since DLTS statespace is uncountable we adopt a finite representation of the DLTS called **region automata** from Alur and Dill [AD94]. The uncountable number of time-vectors representing the different possible combinations of DLTS real-valued clock assignments are represented finitely by a collection of open and closed intervals in the region automata, one interval for each clock, and a relation on clocks that orders them according to the magnitude of the fractional part of the clock value. Hence, the "state" of a region automata consists of a label representing the TSA location, a collection of time intervals, and the fractional-part relation. The intervals and the fractional-part together define equivalence classes for the time vectors. Starting from the initial states, TLCS decides if the mutually reachable set of states satisfy the TLC relation, and it produces counterexamples when TLC does not hold.

3. TIMED LOGIC CONFORMANCE

Based on the bisimulation-style Logic Conformance relation $\succeq_l$ of Stevens [Ste94], we define a timed relation called Timed Logic Conformance $\mathcal{LC}^t$ for Timed Safety Automata (TSA) based on DLTS semantics. $\mathcal{LC}^t$ enforces a time-interval-based relationship between imp actions and spec actions. It also maintains $\succeq_l$'s partial order relationship between specs and imps. $\mathcal{LC}^t$ loosens the standard bisimulation-style strict timed-equivalence requirement formalized by Cerans [Č92], Alur, Courcoubetis, Henzinger [ACH94], and others [LY93]. Instead of strict timed-equivalence we define a weaker relation over the statespaces of two systems based on the time intervals when actions are enabled. The relation requires that imp inputs are enabled over a timed superset of spec inputs[2], and that imp outputs are enabled over a timed subset of spec outputs. $\mathcal{LC}^t$ induces a partial order relation (written $\underset{ob}{\preceq_i}$ for both DLTS and their inducing TSA) over $\mathcal{D} \times \mathcal{D}$. For example, $\underset{ob}{\preceq_i}$ relates TSA Imp (I) and Spec (S) such that $I \underset{ob}{\preceq_i} S$ when $I \succeq_l S$ and all outputs of I occur within the time intervals observed for S's outputs and all inputs of S occur within the time intervals observed for I's inputs. $\mathcal{LC}^t$ is different from other loose timed-refinement

[1] An unstable location is a location with an output or internal transition. Consequently, a **stable location** is a location with no output or internal transitions.

[2] Exceptions to the $I \succeq_l S$ half of the $\mathcal{LC}^t$ relationship are allowed under certain circumstances—see output-bound (ob) definition.

relations [Dan92, Č95]. In particular, $\mathcal{LC}^t$ turns around the standard definition that typically requires imp inputs to be a timed subset of spec inputs. This change is motivated by the need to determine when an imp can be used to replace a spec, and it agrees with common sense that argues one cannot safely substitute an imp that does not accept all of the inputs accepted by the spec.

Like $\succeq_l$, we abstract internal behavior into τ-transitions and ignore internal state changes that are matched by staying in an equivalent state. Recall τ is a distinguished element of Act; hatted actions formalize when τ actions may be matched by staying in the same state and passing zero time as follows: **Tau-abstraction : ($\widehat{\alpha}$):**

$$\forall_{\alpha \in Act \cup I\!R}\ \widehat{\alpha} \triangleq \begin{cases} 0, \alpha = \tau \\ \alpha, \alpha \neq \tau \end{cases}$$

To further loosen action-matching requirements, we extend the transition relations of the systems by transitively closing them over certain action sequences.

τ-closure ($P \stackrel{\alpha}{\Longrightarrow}_\tau Q$): A DLTS transition relation $R \subseteq (S \times (Act \cup I\!R) \times S)$ is τ-transitive if

$$P(\stackrel{\tau}{\longrightarrow})^* \stackrel{\sigma}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^* Q \wedge \sigma \in Act \cup I\!R \quad \vee$$
$$P \stackrel{\delta_1}{\longrightarrow} (\stackrel{\tau}{\longrightarrow})^* \stackrel{\delta_2}{\longrightarrow} Q \wedge \sigma = \delta_1 + \delta_2$$

exists in R then $P \stackrel{\sigma}{\longrightarrow} Q$ also exists in R.

The τ-closure of a DLTS transition relation $R \subseteq (S \times (Act \cup I\!R) \times S)$, is the relation R' such that

1. R' is τ-transitive.
2. $R' \supseteq R$.
3. For any τ-transitive relation R'', $R'' \supseteq R \Rightarrow R'' \supseteq R'$.

The predicate $P \stackrel{\alpha}{\Longrightarrow}_\tau$ is true when there is at least one α-transition from state P. No actions are time abstracted in τ-closure, but it extends transition relations and ignores internal actions that do not matter.

Input-δ-τ-closure ($P \stackrel{\alpha}{\Longrightarrow}_i Q$): In addition to those transitions added to R by τ-closure, input-δ-τ-closure adds a $P \stackrel{\sigma}{\longrightarrow} Q$ transition to R whenever there is a transition sequence $P \stackrel{\delta_1}{\longrightarrow} \stackrel{\sigma}{\longrightarrow} \stackrel{\delta_2}{\longrightarrow} Q$ and $\sigma \in \mathcal{A}, \delta_1, \delta_2 \in I\!R$. The predicate $P \stackrel{\alpha}{\Longrightarrow}_i$ is true when there is at least one α-transition from state P. Input-δ-τ-closure time-abstracts inputs, and extends spec transition relations to match imp inputs.

Output-δ-τ-closure ($P \stackrel{\alpha}{\Longrightarrow}_o Q$): In addition to those transitions added to R by τ-closure, output-δ-τ-closure adds a $P \stackrel{\sigma}{\longrightarrow} Q$ transition to R whenever there is a transition sequence $P \stackrel{\delta_1}{\longrightarrow} \stackrel{\sigma}{\longrightarrow} \stackrel{\delta_2}{\longrightarrow} Q$ and $\sigma \in \overline{\mathcal{A}}, \delta_1, \delta_2 \in I\!R$. The predicate $P \stackrel{\alpha}{\Longrightarrow}_o$ is true when there is at least one α-transition from state P. Output-δ-τ-closure time-abstracts outputs and extends imp transition relations to match spec outputs.

In addition to the closures, we define the following two projections, which are subsets of the DLTS transition relation $\longrightarrow$. They define the sets of spec and imp time-passing actions that must be subsets of each other's time actions.

Input projection ($\circ\!\!-\!\!\!\!\rightarrow_i \subseteq S \times I\!R \times S \triangleq$):

$$\{\langle\langle l, \pi_i\rangle, \delta, \langle l, \pi_j\rangle\rangle \mid \exists \langle\langle l, \pi_k\rangle, \alpha, \_\rangle \in \longrightarrow [\pi_i \leq \pi_k \wedge \pi_j \leq \pi_k \wedge \alpha \in \mathcal{A} \cup \{\tau\}]\}$$

Output projection ($\circ\!\!-\!\!\!\!\rightarrow_o \subseteq S \times I\!R \times S \triangleq$):

$$\{\langle\langle l, \pi_i\rangle, \delta, \langle l, \pi_j\rangle\rangle \mid \exists \langle\langle l, \pi_k\rangle, \beta, \_\rangle \in \longrightarrow [\pi_i \leq \pi_k \wedge \pi_j \leq \pi_k \wedge \beta \in \bar{\mathcal{A}} \cup \{\tau\}]\}$$

Next, we define a predicate that relaxes the superset relationship between imp and spec inputs when simultaneous inputs and outputs are possible from a spec location.

Output-bound (ob):

$ob : S_I \times I\!R \times S_S \times \wp(\langle S_I \times S_S\rangle) \longrightarrow \{t, f\} \triangleq$

$ob(I, \delta, S, \mathcal{R}) \Leftrightarrow$

$$I \stackrel{\delta}{\not\Longrightarrow}_o \wedge \quad (4)$$

$$\exists_{\delta_1 \in I\!R, I' \in S_I, \beta \in A \cup \{\tau\}} [I \stackrel{\delta_1}{\Longrightarrow}_o I' \wedge \beta \in \Psi(I') \wedge \quad (5)$$

$$\forall_{\delta_2 \geq \delta_1, S' \in S_S, I'' \in S_I} [S \stackrel{\delta_2}{\longrightarrow} S' \Rightarrow (I' \mathcal{R} S' \wedge \quad (6)$$

$$(I \stackrel{\delta_2}{\longrightarrow} I'' \Rightarrow I'' \mathcal{R} S'))]] \quad (7)$$

Conjunct 4 requires that the imp cannot do δ. Conjunct 5 requires the imp system to be constrained by an invariant to produce an output or τ. Conjunct 6 insures that future spec actions are matched by the imp when it produces the output, and conjunct 7 specifies that there are no other future imp locations that do not also match the spec's behavior (bisimulation).

The notion output-bound formalizes is that as long as an imp's output occurs within the bounds of the same output in the spec, it can occur in accordance with a stronger location invariant even though the spec could remain in its location longer and subsequently accept future inputs. Without this exception, we generally cannot accept imps with less output variation in locations where otherwise unconstrained inputs are also possible. Modeling locations with both inputs and outputs possible is important for accurate modeling of real systems as well as abstracting behavior into simpler machines with fewer locations.

A **Timed Logic Conformation** ($\mathcal{LC}^t \subseteq S_I \times S_S$) is a binary relation over DLTS automata states between an imp DLTS $\langle S_I, Act_I, \longrightarrow_I, \langle l_0, \pi_0\rangle_I\rangle$, $I, I' \in S_I$ and spec DLTS $\langle S_S, Act_S, \longrightarrow_S, \langle l_0, \pi_0\rangle_S\rangle$, $S, S' \in S_S$ iff

$\forall\ I \mathcal{LC}^t S, \alpha \in \mathcal{A}, \beta \in \bar{\mathcal{A}} \cup \{\tau\}, \delta \in I\!R[$

$$S \stackrel{\alpha}{\longrightarrow} S' \Rightarrow I \stackrel{\alpha}{\Longrightarrow}_o I' \wedge I' \mathcal{LC}^t S' \wedge \quad (8)$$

$$S \stackrel{\beta}{\longrightarrow} S' \Rightarrow I \stackrel{\widehat{\beta}}{\Longrightarrow}_o I' \wedge I' \mathcal{LC}^t S' \wedge \quad (9)$$

$$(I \stackrel{\alpha}{\longrightarrow} I' \wedge S \stackrel{\alpha}{\Longrightarrow}_\tau) \Rightarrow S \stackrel{\alpha}{\Longrightarrow}_i S' \wedge I' \mathcal{LC}^t S' \wedge \quad (10)$$

$$I \stackrel{\beta}{\longrightarrow} I' \Rightarrow S \stackrel{\widehat{\beta}}{\Longrightarrow}_i S' \wedge I' \mathcal{LC}^t S' \wedge \quad (11)$$

$$S \circ\!\!-\!\!\!\!\rightarrow^{\delta}_i S' \Rightarrow ((I \stackrel{\delta}{\Longrightarrow}_o I' \wedge I' \mathcal{LC}^t S') \vee ob(I, \delta, S, \mathcal{LC}^t)) \wedge \quad (12)$$

$$I \circ\!\!-\!\!\!\!\rightarrow^{\delta}_o I' \Rightarrow S \stackrel{\delta}{\Longrightarrow}_i S' \wedge I' \mathcal{LC}^t S'] \quad (13)$$

Formulas 8 and 9 require that the imp simulates the observable behaviors of the spec. Formulas 10 and 11 require that the spec simulate observable behaviors of the imp. Formula 10 weakens standard weak bisimulation allowing imps to have **irrelevant inputs**; i.e. inputs the spec does not accept in state S or its τ-derivatives. This can save considerable time when computing TLC. Formula 11 requires the spec to simulate all imp outputs and τs. Formula 12 insures that the imp simulates all spec-input time derivatives with output-bound exceptions allowed, and Formula 13 insures that the spec simulates all imp-output time derivatives.

We introduce the following TSA modeling constraints to insure that TLC is transitive over the statespace of DLTSs induced from constraint-satisfying TSA:

1. A location l has an invariant ρ_l iff it is a from-location for one or more output or tau transitions.
2. No upper bounded guards exist in output or tau transition guards. Inputs may have such guards.
3. No to-location of a transition has a stronger invariant than the from-location unless the strengthened-invariant-clause clock is reset.

These are reasonable modeling constraints—especially for the hardware domain where devices do control the occurrence of their outputs but not their inputs. The constraints strengthen the causal relationship of the models and their outputs and they increase the fidelity between the models and the physical devices they represent. Under these modeling constraints, the TLC relation is a partial order with respect to weak timed bisimulation equivalence between DLTSs [Youss].

We must narrow down the definition of $\mathcal{LC}^t$ so that it uniquely defines one of the many possible relations between DLTS states (e.g. $\emptyset$ is a useless $\mathcal{LC}^t$). The $\mathcal{LC}^t$ relation we desire is the union of all subsets of $S_I \times S_S$ that are timed logic conformations, or $\mathcal{LC}^t$'s **maximum fixpoint** denoted $\widehat{\mathcal{LC}^t}$. We can safely substitute an imp DLTS I for a spec DLTS S when their initial states are in $\widehat{\mathcal{LC}^t}$.

An imp TSA I is **timed logic conformant** to spec TSA S (written $I \underset{\circ}{\leftrightarrows}_i S$) whenever the DLTSs induced from I ($I' = \langle S_I, Act_I, \longrightarrow_I, \langle l_0, \vec{0}\rangle_I\rangle$) and S ($S' = \langle S_S, Act_S, \longrightarrow_S, \langle l_0, \vec{0}\rangle_S\rangle$) are timed logic conformant (written $I' \underset{\circ}{\leftrightarrows}_i S'$). A pair of DLTS automata are timed logic conformant to each other when their initial states are in the maximum fixpoint TLC state relation over their statespaces; i.e.:

$$I' \underset{\circ}{\leftrightarrows}_i S' \triangleq \langle\langle l_0, \vec{0}\rangle_I, \langle l_0, \vec{0}\rangle_S\rangle \in \widehat{\mathcal{LC}^t}$$

4. HIERARCHICAL VERIFICATION

Top-down hierarchical TLC-verification starts at the most abstract level with a spec that incorporates the environmental timing issues (e.g. input frequency, stimulus-response constraints) into its behavior. The spec is our contract with the environment; as such it defines the behavior required for the inputs it accepts. Only imps that satisfy the TLC relation with the spec fulfill the contract; TLC failures are design errors. A parallel composition with an output offered in a non-accepting state is a design error called *computation interference* (CI). A composition is *CI-free* when all receivers accept every output offered by transmitters in the composition's reachable statespace. All non-parallel TSA are CI-free by definition. A top-level parallelly-composed spec must be CI-free.

A hierarchical system can be top-down verified by defining (usually by parallel composition) a set of sub-specs that are TLC-verified against the spec. Sub-specs must also be CI-free, but only in the subset of their statespace explored by the TLC-relation with the spec's reachable statespace. We continue down the hierarchy TLC-verifying each sub-spec against its sub-sub-spec until TLC holds with imps composed entirely of design primitives. The reverse method can be used from the bottom-up to create systems (as done in the STARI example [Youss]). We believe The CI-free property preserves the TLC relation across parallel composition and frees us from having to specify behaviors for all possible inputs in all possible states for all possible times. This greatly simplifies the modeling and verification task, and directly supports abstractly verifying components across levels of hierarchy.

5. APPLICATION

The flexible time and behavior modeling capabilities of Timed Safety Automata (TSA) express the relationship between time passing and behavior at many different levels of abstraction. We introduce two canonical forms for modeling logic gates; the first is called **monotonic**. A monotonic model reflects every possible output change that can occur from all unstable locations. The simple inverter shown earlier in Figure 1 is monotonic. In contrast, an **inertial** model accepts stabilizing inputs in unstable states and will not reflect an output change when a stabilizing input occurs. An inertial inverter is identical to the TSA in Figure 1, except that it includes two additional a-transitions, 00⟶10 and 00⟶10, guarded by k<MinD. A spike on the a input to the inertial inverter may occur and not generate a b_ output action; this inverter has inertial-delay semantics during the interval [0,MinD). In practice, it might be the case that an even smaller inertial time period, and not the whole time interval [0,MinD) would be better for high fidelity modeling, and such a model can be accommodated by adding another timing parameter to the TSA, but for simplicity, and in agreement with the general bi-bounded delay model [BS91, Bur92], we will not describe more detailed models in this paper.

5.1. Simple Nand-Inverter And-gate Example

Figure 2 depicts the logic symbol and the TSA defining the behavior of an inertial two-input and-gate. In this model, locations are labeled with three-digit binary codes indicating the values of the and-gate's two inputs and output in that location. For example, the location 101 is an unstable location, where input a is asserted, and input b is de-asserted, and output c_ is asserted. Every TSA input from a stable to unstable location resets k and every unstable location has the invariant $k \leq$ MaxD for some integral delay MaxD. The and-gate can start from any stable location.

A nand-gate is very similar to an and-gate, except that the and-gate stable/unstable locations are nand-gate unstable/stable locations. Hence, the location invariants are

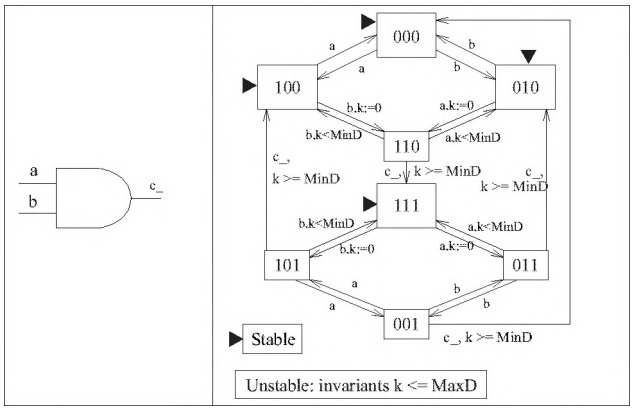

Figure 2. Two-Input And-Gate

swapped from the unstable and-locations to the unstable nand-locations, and transitions are reversed.

One and-gate imp is a coupled nand-gate and inverter as shown at the top of Figure 3. Depending on the timing of the gates, this parallel and-gate is an acceptable imp of the and-gate "spec" in Figure 2. It is interesting to compare the

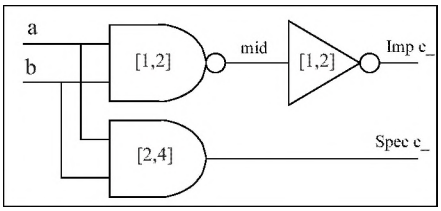

Figure 3. And-gates in Parallel

timing relationships TLC accepts. Generally, given and-gate's minimum and maximum delays AndMin and AndMax, one expects that the timing relationship is satisfied whenever NandMin + InvMin $\geq$ AndMin and NandMax + InvMax $\leq$ AndMax. That is the case when monotonic gates are used, but, that is not the case with inertial gates! The parallel-and imp can output a `c_` earlier than the and-gate spec allows when NandMin + InvMin = AndMin with inertial gate models! For example, assume that the nand-gate and inverter minimum and maximum delays are 1 and 2 time units, and that the and-gate spec minimum and maximum delay is 2 and 4 time units respectively. Imagine the imp and spec inputs wired in parallel together as diagramed in Figure 3, and refer to the timing diagram in Figure 4.

Let T be our point of reference for time passing, and let $T = 0$ just when the last input is asserted from 0 to 1. It is possible then that at $T = 1.5$ the TSA can be in locations Imp:[nand111, inv10], and Spec:and110, where both the Imp and Spec are in unstable locations. Then, τ(`mid_`) de-asserts moving to locations Imp:[nand110, inv00], and Spec:and110 where only the nand-gate TSA is in a stable location. If another `a` input occurs at $T = 1.75$, before the and-gate can assert its output, then the and-gate spec stabilizes in state and010, and the nand-gate destabilizes to nand010. Eventually, if no more inputs occur before $T = 3.75$, the nand-gate will assert τ(`mid_`)and stabilize in state nand011. But until it does, the inverter is still unstable in state inv00, and it can generate a `c_` output

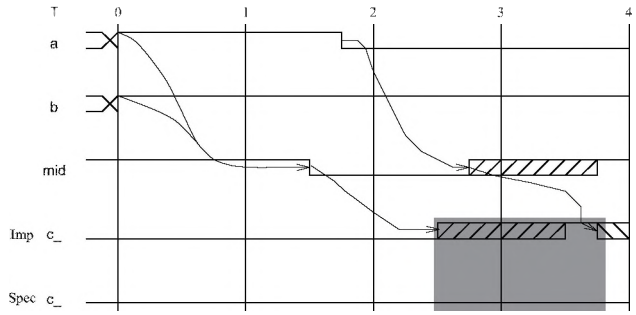

Figure 4. And-gate Imp-Spec Timing Diagram

for $T \in [2.5, 3.5]$. The spec cannot generate this `c_` output. This difference between the spec and imp outputs is highlighted by the shaded area in Figure 4. If however we change timing parameters such that the spec delay is $[1, 4]$ (or some superset of $[1, 4]$), and the nand and inverter delays are $[1, 2]$, TLC is satisfied because the imp cannot then produce any outputs outside the time bounds allowed by the spec. Discovering problems at delay boundaries like this is the key to building reliable circuits.

In general, for inertial gates with non-zero gate delays, given that PMin $\triangleq$ NandMin + InvMin and PMax $\triangleq$ NandMax + InvMax, TLC holds whenever PMin $\geq$ AndMin $\wedge$ PMax $\leq$ AndMax $\wedge$ AndMin $\leq$ NandMin.

Verification results are very dependent on the models chosen as illustrated in this example. In particular, TLC verification results are different for the monotonic and inertial gate models. Some of the most difficult errors to track down in hardware devices are those associated with unstable states and their upper and lower delay bounds; since we are interested in creating designs that do not suffer from obscure defects like this, we recommend verifying with inertial gate models. Discovering when there are dependencies in real designs that keep these kind of problems from occurring is the key to building hierarchical systems that are efficient. Both are supported by TLC verification with appropriately detailed models.

5.2. STARI Queue and Perfect Buffer

STARI (Self Timed at Receiver's Input) is an asynchronous queue designed fabricated by Greenstreet [Gre93]. STARI connects two systems that operate at the same clock rate but with some clock skew. The general problem is to size the queue such that it buffers the data between the clock-skewed systems allowing the transmitter to output and the receiver to input a new data value every clock cycle without waiting for the queue. We hierarchically model and verify STARI operation against a perfect buffer spec in [Youss]. Generally, we model systems in more detail than the other STARI queue verifications we are aware of [Gre93, BM98, TB97], and we are able to compute comparable results. Using the asynchronous STARI verification problem as a benchmark, we confirm Berkeley researchers results [TB97]; we extend the verification to include data values passing correctly through the queue and we do not use assumes-guarantees reasoning to accomplish the verification. Comparing our work with French researchers from VERIMAG [BM98], we generally confirm their results but

point out an important counterexample when set-up and hold time requirements of the receiver are taken into account. Our model is much more detailed than the original proof of STARI correctness [Gre93], proving properties about the actual data transferred as well as showing a counterexample to the formula derived for allowable skew between sender and receiver clocks.

6. CONCLUSION

TSA are well suited for modeling systems at various levels of abstraction, and the TLC relationship is useful for verifying when one TSA is an acceptable implementation of another. TLC verification is an alternative to assumes-guarantees-based verification. TSA allow one to naturally incorporate the environmental constraints into specs and the TLC decision procedure insures those environmental constraints are satisfied by acceptable implementations. The environmental constraints restrict the number of states that must be examined and they make a fair tradeoff possible between model fidelity and computational complexity. Once imp/spec pairs satisfy TLC, the spec can be safely and efficiently substituted for the imp in higher-level verifications without reaccomplishing assumes-guarantees proof obligations. TLC helps efficiently build high-performance hierarchical systems by discovering timing dependencies in real designs that keep real hazards safely under control.

Our contributions are:

- A formal definition of a practical relationship that designers can use to decide when an imp satisfies a spec.
- A verification process that supports using more detailed models and discovers more problems because the TLC relation naturally restricts the verification to state-pairs that matter.
- A simple verification process where the environment constraints are modeled in the spec rather than in separate models of the environment. This avoids assumes-guarantees proof-obligations that aren't part of the equivalence checking process itself and that must be reaccomplished when the environment model changes or a connected part of the design changes.
- Canonical inertial and monotonic modeling techniques.
- A formal definition for TSA parallel composition and a procedure implementing it.
- An automated decision procedure for computing the TLC relation and generating counterexamples.
- STARI verification using more detailed models to expose potential problems not revealed by others.

REFERENCES

[ACH94] R. Alur, C. Courcoubetis, and T. Henzinger. The observational power of clocks. In *Proceedings of CONCUR '94*. LNCS 836, 1994.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[BM98] Marius Bozga and Oded Maler. Modeling and verification of the stari chip using timed automata. In *Proceedings of Cav '98*, 1998.

[BS91] J.A. Brzozowski and C-J. H. Seger. Advances in asynchronous circuit theory part ii: Bounded inertial delay model, mos circuits, design techniques. In *EACTS Bulletin 43*, pages 199–263. 1991.

[Bur92] Jerry Burch. Delay models for verifying speed-independent asynchronous circuits. In *Proceedings of the International Conference of Computer Design (ICCD)*, pages 270–274. IEEE Computer society Press, oct 1992.

[Dan92] Mats Daniels. Modelling real-time behavior with an interval time calculus. In *Formal Techniques in Real-Time and Fault-Tolerant Systems. Second International Symposium Proceedings*, 1992.

[Gre93] Michael R. Greenstreet. *STARI: A Technique for High-Bandwidth Communication*. PhD thesis, Princeton, January 1993.

[HNSY94] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.

[LY93] Kim G. Larsen and Wang Yi. Time-abstracted bisimulation: Implicit specifications and decidability. In *Proceedings of Mathematical Foundations of Programming Semantics (MFPS)'93*, pages 160–176, 1993.

[SS95] Oleg V. Sokolsky and Scott A. Smolka. Local model checking for real-time systems. In *Proceedings of CAV'95*, 1995.

[Ste94] Kenneth S. Stevens. *Practical Verification and Synthesis of Low Latency Asynchronous Systems*. PhD thesis, The University of Calgary, Calgary, Alberta Canada, September 1994.

[TAKB96] Serdar Tasiran, Rajeev Alur, Robert P. Kurshan, and Robert K. Brayton. Verifying abstractions of timed systems. In *Proceedings of 7th International Conference on Concurrency Theory*, pages 546–562. Springer-Verlag, 1996.

[TB97] Serdar Tasiran and Robert K. Brayton. Stari: A case study in compositional and hierarchical timing verification. In *Proceedings of the Computer Aided Verification Conference*, 1997.

[Č92] K. Čerāns. Decidability of bisimulation equivalences for parallel timer processes. In *Proceedings of CAV '92*. LNCS 663, 1992.

[Č95] Kārlis Čerāns. Ctr: A calculus of timed refinement. In I. Lee and S. Smolka, editors, *Proceedings of CONCUR '95*, pages 516–630, 1995.

[Youss] Frank C. D. Young. *Timed Safety Automata and Logic Conformance*. PhD thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH 45433, 1999 (In Progress).

TIMED LOGIC CONFORMANCE AND ITS APPLICATION

Frank C. D. Young[1] , *Kenneth S. Stevens*[2] *and Robert P. Graham Jr.*[3]

[1]Air Force Research Laboratory, Wright-Patterson AFB, OH 45433, USA

[2]SCL, Intel Corporation, Hillsboro OR, 97124, USA

[3]Air Force Institute of Technology, Wright-Patterson AFB, OH 45433, USA

Timed Logic Conformance (TLC) is a bisimulation-style partial order relationship defined over the statespace of Timed Safety Automata (TSA) with real-valued clocks. In contrast to timed simulation, Calculus of Timed Refinement (CTR), and Time-Abstracted bisimulation, TLC defines when one system is an acceptable implementation of another by asymmetric bisimulation-style requirements for specification inputs and implementation outputs. While TLC does not necessarily preserve timed properties, it intuitively and pragmatically supports writing abstract specifications and verifying them against implementations. TLC scales up by substituting verified specifications for implementations and hierarchically verifying larger systems. TLC verification is an alternative to assumes-guarantees reasoning process. TLC verification depends on explicitly capturing environmental timing properties in the specification and insuring they are satisfied in the TLC relation. The region-automata-based TLC System (TLCS) implements TSA parallel composition and a TLC decision procedure which is used to hierarchically verify the STARI queue.