Asynchronous QDI Circuit Synthesis from Signal Transition Protocols

Bo-Yuan Huang1, Yi-Hsiang Lai2, Jie-Hong R.Jiang1,2
1Department of Electrical Engineering, National Taiwan University, Taipei, 10617, Taiwan
2Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, 10617, Taiwan

ABSTRACT
Asynchronous circuits are promising in resolving the emerging issue of process variation and high synchronization power consumption. Among various asynchronous delay models, quasi-delay insensitive (QDI) model is the most robust and yet practical one due to its relaxed timing assumption. However, automatic synthesis of QDI circuits from signal transition graph (STG) protocol specification has not yet been proposed, despite the fact that algorithms synthesizing circuits under other delay models do exist. In this paper we propose the first algorithm synthesizing protocols specified in STGs into QDI circuits by analyzing STG structures without utilizing state graph assignment techniques. Furthermore, an optimization technique is proposed to simplify QDI circuits. In our synthesis algorithm, the state explosion issue is avoided, and restrictions on STGs are relaxed. Case studies on Advanced Microcontroller Bus Architecture (AMBA) and other protocols indicate the feasibility of our method.

1. INTRODUCTION
Asynchronous techniques gain their application in modern integrated circuit design. The rapid growth of system complexity makes synchronizing all flip flops in a chip difficult, and the variability of integrated circuit manufacturing makes the design-for-the-worst-case methodology inadequate. Synchronizing clock signals and controlling gate and interconnect delays within a chip can lead to substantial power consumption. Due to the promising strengths of asynchronous circuits in addressing these issues, myriad research efforts have been contributed; see, e.g., [1] for a brief history of asynchronous design.

Several asynchronous delay models have been proposed that vary in their underlying timing assumptions, such as delay insensitive (DI), quasi-delay insensitive (QDI), speed independent (SI), and other models. Among these delay models, the QDI model is the most robust and yet practical one. While the SI model assumes unbounded gate delay and zero wire delay, the QDI model has almost no timing assumption except for the isochronic fork assumption. QDI circuits assume the difference in time at which a signal arrives at each end of an isochronic fork is less than the minimum gate delay. This assumption can be easily met if the designated wires are physically localized in circuits. Due to their robustness, QDI circuits can be designed and synthesized in a way similar to the synchronous design flow [2–5].

Algorithms for control protocol synthesis are essential due to the thriving needs for system-level circuit designs. While signal transition graph (STG) is regarded as a specification formalism for asynchronous control protocols [6], many researchers studied asynchronous circuit synthesis from STGs. Also there are asynchronous synthesis tools utilize STGs to improve their performance [7]. Among the proposed STG synthesis algorithms, many solved the problem through state graph (SG) assignment, which requires generating intermediate SGs [6, 8, 9]. However, this approach would inevitably encounter the state explosion issue because the size of the SG is potentially exponential to the number of signals in the STG. Several methods such as STG decomposition [10, 11], STG unfolding [11, 12], and formulation with integer linear programming (ILP) [13] were proposed to tackle this issue. Other synthesis algorithms utilized graph-based approach to avoid constructing intermediate SGs [14–17]. Disregarding the state explosion issue, previous algorithms were only applicable on STGs with complete state coding (CSC) property. Furthermore, the property persistency is also required for SI circuits [18]. Several methodologies aiming for efficient verification/modification for CSC and persistency property were proposed [12, 19]. Apart from the extra cost for validity verification, the persistency property of STGs might reduce concurrency and degrade circuit performance due to its additionally imposed signal transition constraints. Control protocol synthesis from other specifications were also proposed [20].

In order to synthesize STGs into QDI circuits without the requirements of CSC and persistency property, we define five behavior models and propose a method to analyze the topological relation between signal transitions in single-cycle STGs. Every output transition and its enabling transitions can be completely classified into the five behavior models. We use triggering condition (TC) to describe the conditions for a signal to change its value concerning logic levels of input signals; triggering pattern (TP) for each behavior model is derived to compose TCs for signals in simply-driven STGs. Based on the TCs we derived, output signals synthesized are effective, stable and precise under the QDI assumption. We also propose optimization techniques to reduce circuit cost by restricting system concurrency. Overall, this synthesis algorithm has complexity \(O(|E|^2)\), where \(|E|\) is the number of edges of the underlying STG. Experimental results suggest that our method can be effective in synthesizing circuits from practical protocols, such as the AMBA specification.

2. PRELIMINARIES
A Petri net [21] is a tuple \((Q,F,T,M_0)\), where \(Q\) is a set of places, \(T\) is a set of transitions, \(F \subseteq (Q \times T) \cup (T \times Q)\) defines the relation flow among \(Q\) and \(T\), and \(M_0\) represents the initial marking. If \((t,q) \in F\) then \(t \in T\) is an input transition of \(q \in Q\); if \((q,t) \in F\) then \(t \in T\) is an output transition of \(q \in Q\). A marking is a set of non-negative integers that represents the number of tokens on each place. A transition \(t\) is enabled and may be fired if all of its input places have at least one token. We also call a transition is triggered or occurs if it is fired. Note that a transition may not be immediately triggered even if it is enabled. We denote \(Reach(M_0)\) as the set of reachable markings given initial marking \(M_0\).

A path \(P\) is an ordered set of transitions that represents a part of firing sequence. If transition \(u^*\) is fired within path \(P\), we denote it as \(u^* \in P\). A cycle is a path with the same starting and ending transition. A path or cycle is called simple if all transitions in it appears exactly once. In this paper, we focus on a special Petri net, called a marked

---

978-1-4673-8388-2/15/$31.00 ©2015 IEEE 434
where every place has exactly one input transition and one output transition. A marked graph is live and safe if it is strongly connected and every place contains no more than one token within any marking reachable from the initial marking [16]. Liveness of STGs guarantees no deadlock occurs [18]. In addition, a marked graph is simple if its initial marking $M_0$ places exactly one token on each simple cycle [6, 16]. Since a valid STG is an interpreted simple, live and safe marked graph by Definition 1, the existence of the two paths are therefore implied.

**Definition 1.** A signal transition graph (STG) $G = (V, E)$ is an interpreted subset of marked graph wherein each transition represents either the rising ($x^+$) or falling ($x^-$) of a signal $x$, which has signal levels high and low. $V$ is the set of transitions and $E$ is the set of edges corresponding to places of the underlying marked graph.

We denote the set of signals as $S$, the set of input signals as $PI$ and the set of output signals as $PO$, that is to say, $S = PI \cup PO$ and $PI \cap PO = \emptyset$. We define $x^*$ as an arbitrary transition of signal $x$, which is either $x^+$ or $x^-$. We call an STG valid if its underlying marked graph is simple, live and safe. Taking T-element [22] in Figure 1(a) for instance, $PI = \{\text{lr, Or}\}$, $PO = \{\text{la, Or, Or}\}$ and $S = \{\text{lr}, \text{la, Or, Or}\}$. We label transitions of input and output signals in red and black, respectively, in all the following figures.

![Figure 1: T-element (a) IO block (b) STG](image)

**Definition 2.** A signal is called single-cycle if there is exactly one rising transition and one falling transition within the STG. Otherwise, it is called a multi-cycle signal. A single-cycle STG contains only single-cycle signals.

In this paper, we focus on single-cycle STGs to study the relation between STG structure and signal transitions. In a single-cycle STG, the rising transition $x^+$ and falling transition $x^-$ of a signal $x$ are designated as the conjugate transition of each other and is denoted by $\lnot$, e.g., $x^+ = \lnot x^-$. 

**Definition 3.** Two transitions $x^*, y^*$ are concurrent, denoted as $(x^* \parallel y^*)$, if the two transitions are both enabled within some markings reachable from the initial marking $M_0$. In contrast, two transitions $x^*, y^*$ are sequential, denoted as $(x^\sim y^*)$, if the two transitions are not enabled simultaneously within any marking reachable from the initial marking $M_0$.

If two transitions are sequential, triggering order between them can be defined given a reference transition. We write $(u^* \gg x^* \gg y^*)$ to denote the cases where transition $x^*$ must be triggered before transition $y^*$ fires once transition $u^*$ has been triggered. For instance, in Figure 1(b), $(Or^+ \gg \text{la}^+) \parallel (\text{la}^- \gg \text{la}^-)$.

We denote $P_{x^*}(x^*, y^*)$ as a set of simple paths that start from $x^*$ and end with $y^*$, i.e., $P_{x^*}(x^*, y^*) = \{P \mid P = \{x^*, \cdots, y^*\}\}$. We define the symbol $\xi$ as $u^* \in P_{x^*}(x^*, y^*) \Rightarrow \exists P \in P_{x^*}(x^*, y^*)$ s.t. $u^* \in P$. For example, there exists a simple path $P = \{\text{Or}^+, \text{la}^+, \text{lr}^-, \text{la}^-\}$ in Figure 1(b) and $P \in P_{x^+}(\text{Or}^+, \text{la}^-)$. Also, $Or^- \in P_{x^+}(\text{Or}^+, \text{la}^-)$ and $\text{lr}^+ \notin P_{x^+}(\text{Or}^+, \text{la}^-)$.

**Lemma 1.** In a valid STG, $(x^* \sim y^*) \Leftrightarrow \exists P_1 \in P_{x^*}(x^*, y^*) \land \exists P_2 \in P_{y^*}(y^*, x^*)$ such that the cycle formed by $P_1$ and $P_2$ is simple.

**Proof.** Within a simple, live and safe marked graph, two transitions are concurrent if and only if they do not reside on a simple cycle [6, 16]. Since a valid STG is an interpreted simple, live and safe marked graph by Definition 1, the existence of the two paths are therefore implied.

**Lemma 2.** In a valid and single-cycle STG, every transition must be sequential to its conjugate transition, i.e., $\forall u \in S(u^+ \sim u^-)$.

**Proof.** If there exists $u \in S$ with $(u^+ \parallel u^-)$, then there exists $M_1 \in \text{Reach}(M_0)$ such that both $u^+$ and $u^-$ are simultaneously enabled under $M_1$, which will cause ambiguity in signal level.

The following corollary follows from the above two lemmas.

**Corollary 1.** Given a valid and single-cycle STG, there exists a path connecting conjugate transitions of each signal, i.e., $\forall u \in S \exists P_1 \in P_u(u^+, u^-) \land \exists P_2 \in P_u(u^-, u^+)$.  

We define the enabling set $En(t^*)$ of a transition $t^*$ to be the set of input transitions, i.e., $En(t^*) = \{u^+ \mid (u^+, t^+) \in E\}$. Taking T-element in Figure 1(b) for example, $En(\text{la}^+) = \{\text{Or}^+\}$ and $En(\text{la}^-) = \{\text{lr}^-, \text{Or}^-\}$.

### 2.1 STG Properties and Handshake Protocols

We focus on STGs containing only input and output signals. Because internal signals are not necessary in representing control protocols and, nevertheless, they can be transformed to dummy outputs or be eliminated.

Due to the characteristics of QDI circuits, we focus on valid STGs whose input transitions and output transitions interleave. The reasons are twofold. First, the order of consecutive output transitions is not very meaningful under the QDI model, where wire delays are unbounded and the environment may thus receive transitions in different order. Second, edges in an STG indicate the causality between transitions, thus input transitions should be triggered only by output transitions. Our assumption on IO-interleaving is reasonable for QDI circuits. Furthermore, with IO-interleaving, the synthesized circuits have no internal feedback, which may result in signal metastability.

**Definition 4.** In a valid, single-cycle, and IO-interleaving STG, an input $u$ is simply-driven if $u^* \in En(x^*)$, $u^* \in En(x^-)$, and $\forall v^* \in En(x^-), u^* \in En(x^+)$. An STG is simply-driven if it is a valid, single-cycle, and IO-interleaving STG with all input signals being simply-driven.

For a simply-driven (SD) STG, input transitions change their signal levels with respect to certain output transitions. It is similar to the request-acknowledge handshake mechanism. Taking T-element in Figure 1(b) for instance, $En(\text{lr}^*) = \{\text{la}^-\}$ and $En(\text{lr}^+) = \{\text{la}^+\}$. Input signal $\text{lr}$ changes its signal level depends only on the output it received, say signal $\text{la}$, and it rises and falls under complementary conditions. This paper focuses on simply-driven STGs. With our case study on AMBA protocol and empirical experience, we believe this assumption is practical for real-life protocols under QDI realization.

### 3. STG STRUCTURE ANALYSIS

In this section, we propose a formal method to analyze relations between signal transitions and define five behavior models to represent topological relations between output transitions and their enabling transitions. In the following discussion, we assume $u^* \in En(x^*)$.\(^{(1)}\)

We study the cases whether $u^*$ is concurrent or sequential to $x^*$ or $x^-$, and we categorize the topological orders of related transitions $u^*$, $u^-$, $x^*$, and $x^-$ into the Staggered, Back-to-Back, Sneaking, Liable, and Independent models. Each model will be characterized with two equivalent forms: topological relations and path constraints.
3.1 Liable

The transitions $u^*$ and $x^*$ in this model are sequential with $x^-$ after $x^*$ and before $x^-$. The topological relation of this model is

$$(u^- \sim x^*) \land (u^- \sim x^-) \land (x^* \Rightarrow u^- \Rightarrow x^-).$$

The transitions $u^*$ and $x^*$ belong to the Liable model if they satisfy the path constraint

$$u^- \in \mathcal{P}(x^*, x^-) \land
(u^- \notin \mathcal{P}(x^-, x^*) \lor \forall P \in \mathcal{P}(x^*, x^-), u^- \in P).$$

3.2 Back-to-Back

Similar to the Liable model, transitions $u^*$ and $x^*$ in this model are mutually sequential. However, transition $u^-$ follows $x^-$ once $x^*$ is triggered. The topological relation of this model is

$$(u^- \sim x^*) \land (u^- \sim x^-) \land (x^* \Rightarrow x^- \Rightarrow u^-).$$

The transitions $u^*$ and $x^*$ belong to the Back-to-Back model if they satisfy the path constraint

$$u^- \in \mathcal{P}(x^-, x^*) \land
(u^- \notin \mathcal{P}(x^-, x^-) \lor \exists P \in \mathcal{P}(x^-, x^-), u^- \in P).$$

3.3 Sneaking

In this model only $x^-$ is sequential to $u^-$, and $u^-$ fires before $x^-$ once $u^*$ has been triggered. The topological relation of this model is

$$(u^- \parallel x^*) \land (u^- \sim x^-) \land (u^* \Rightarrow u^- \Rightarrow x^-).$$

The transitions $u^*$ and $x^*$ belong to the Sneaking model if they satisfy the path constraint

$$u^- \notin \mathcal{P}(x^*, x^-) \land u^- \notin \mathcal{P}(x^-, x^*) \land
u^- \in \mathcal{P}(x^-, x^*) \land u^- \notin \mathcal{P}(x^-, x^-).$$

3.4 Liable

In this model only $x^*$ is sequential to $u^-$, and transition $u^-$ must occur after $x^*$ once $u^*$ has occurred. The topological relation of this model is

$$(u^- \sim x^*) \land (u^- \parallel x^-) \land (u^* \Rightarrow x^* \Rightarrow u^-).$$

Transitions $u^*$ and $x^*$ belong to the Liable model if they satisfy the path constraint

$$u^- \notin \mathcal{P}(x^*, x^-) \land u^- \notin \mathcal{P}(x^-, x^*) \land
u^- \notin \mathcal{P}(x^-, x^-) \land u^- \in \mathcal{P}(x^*, x^*).$$

3.5 Independent

In this model transition $u^-$ is sequential to neither $x^*$ nor $x^-$. The topological relation of this model is

$$(u^- \parallel x^*) \land (u^- \parallel x^-).$$

Transitions $u^*$ and $x^*$ belong to the Independent model if they satisfy the path constraint

$$u^- \notin \mathcal{P}(x^*, x^-) \land u^- \notin \mathcal{P}(x^-, x^*) \land
u^- \notin \mathcal{P}(x^-, x^-) \land u^- \notin \mathcal{P}(x^*, x^*).$$

With the above categorization, analysis on topological structure can be applied to every transition and its enabling set. Furthermore, either two transitions are sequential or concurrent can be determined by the existence of path between them, which can be efficiently determined by a depth-first traversal.

**Lemma 3.** For valid and single-cycle STGs, the topological relation and the path constraint characterized by each of the above five models are equivalent.

**Proof.** The equivalences between the topological relation and the path constraint under models Sneaking, Liable and Independent follow from Lemma 1.

For models Liable and Back-to-Back, we consider the following two conditions:

$$u^- \in \mathcal{P}(x^*, x^-) \quad (2)$$

$$u^- \in \mathcal{P}(x^-, x^*) \quad (3)$$

In both Liable and Back-to-Back models, at least one of the conditions (2) and (3) holds due to $(u^- \sim x^*) \land (u^- \sim x^-)$, implied by Lemma 1. Transitions that satisfy (2) but not (3) can be clearly classified by the Liable model. Similarly, transitions that satisfy (3) but not (2) can be classified by the Back-to-Back model.

For transitions that satisfy both (2) and (3), there are several cases: If $\forall P \in \mathcal{P}(x^*, x^-)$, $u^- \in P$, then the transitions should certainly be classified with the Liable model since $x^-$ would never be enabled before $u^-$ is fired. If $\exists P \in \mathcal{P}(x^*, x^-)$ such that $u^- \notin P$, the transitions may be enabled in different orders depending on the initial markings as the examples shown in Figure 3, where blue circles indicate initial markings. Note that the differences between Figure 3 (a) and (b) and between (c) and (d) are whether $u^* \in \mathcal{P}(x^-, x^*)$.
Therefore, for a valid and single-cycle STG, the topological relation and the path constraint characterized by each of the behavior model are equivalent. □

**Theorem 1.** Given any valid and single-cycle STG, the above five behavior models completely cover all possible structures among transitions.

**Proof.** Based on Lemma 3, the possible structures among transitions that are covered by the five models can be characterized by the union of the path constraints defined by the five models. Taking the union of the path constraints of the Sneaking, Liable, and Independent models, we have

\[
\begin{align*}
   u^- \in P_a(x, x^-) \land u^- \in P_b(x, x^-) \land \\
   (u^- \in P_a(x, x) \lor u^- \in P_b(x, x^+)).
\end{align*}
\]

(4)

On the other hand, taking the union of the path constraints of the Staggered and Back-to-back models, we have

\[
\begin{align*}
   u^- \in P_a(x, x^-) \lor u^- \in P_b(x, x^+),
\end{align*}
\]

(5)

which is the union of Eq. (2) and Eq. (3). Hence, we only have to consider the rest possible cases that is, both Eq. (2) and Eq. (3) fail and \( u^- \in P_a(x, x^-) \land u^- \in P_b(x, x^-) \).

We show that such cases do not exist. The assumption \( \exists P_a \in P_a(x, x^-) \) with \( u^- \in P_1 \) and \( \exists P_b \in P_b(x, x^-) \) with \( u^- \in P_2 \) may imply the existence of a path \( P \) so that \( P \in P_a(x, x^-) \cup P_b(x, x^-) \), which contradicts to (2) and (3). Therefore such cases are excluded.

As a consequence, the transitions of a valid and single-cycle STG are completely covered by the five behavior models. □

### 4. Triggering Conditions and Triggering Patterns

To derive the conditions for output signals to change their signal levels, we define *triggering condition* (TC) to represent the conditions for a transition to be triggered, and derive *triggering pattern* (TP), which describes how an output signal is triggered. The new STG is conservative to the original STG if no deadlock is caused.

**Definition 5.** In representing triggering condition (TC), we denote \( R_0(x) \) as the pull-up set and \( F_i(x) \) as the pull-down set of signal \( x \). The pull-up and pull-down sets of a signal are sets of transitions whose triggering are necessary for the signal to rise and fall, respectively.

Taking the standard two-input C-element [23] for example, let the input signals be \( a, b \) and output signal \( x \). Then the TC of \( x \) is \( R_0(x) = \{ x^-, b^- \} \) and \( F_0(x) = \{ a^-, b^+ \} \). Note that forks to the pull-up set and to the pull-down set of a same signal are designated isochronic.

In order to derive TC for every output signal, we analyze the behavior model to which every output transition and its enabling transition belong. Based on the model derived, we gradually add elements into TCS by applying TPs, which are expressed in the format:

\[
x^+ = R(u^+)F(t^+).
\]

(6)

It denotes, for some \( u^+ \) and \( t^+ \), adding \( u^+ \) to \( R_0(x) \) and adding \( t^+ \) to \( F_0(x) \) if \( x^+ = x^- \), or adding \( u^+ \) to \( F_0(x^+) \) and \( t^+ \) to \( R_0(x^-) \) if \( x^+ = x^- \). For the C-element, the TC of output \( x \) in terms of TP is expressed as \( x^+ = R(a^+)F(a^-) \) and \( x^- = R(b^+)F(b^-) \).

A signal is called **effective** if it is enabled once every enabling transition has been fired; a signal is called **stable** if there is no glitch between rising and falling transition; a signal is called **precise** if it guarantees no ambiguity between rising and falling transitions.

**Proposition 1.** If, for some signal \( x \), there exists a transition in pull-up set whose conjugate transition is contained in pull-down set, i.e., \( \exists u^+ \in V \) such that \( u^+ \in R_0(x) \lor u^+ \in F_0(x) \), then the signal \( x \) is precise.

Implementing an STG with all behaviors being preserved may be at the expense of circuit complexity. Since reduction on system flexibility may lead to improvement in circuit cost, we define the conservative property below to constrain circuit synthesis with system flexibility.

**Definition 6.** A valid STG \( G_1 \) is conservative to a valid STG \( G_2 \) if the system behavior, a set of signal transition traces, of \( G_1 \) is a subset of that of \( G_2 \).

**Lemma 4.** Adding one edge on a valid STG guarantees it to be conservative to the original STG if no deadlock is caused.

**Proof.** Consider adding an edge \( E = (Y, X) \) on a valid STG, where \( X \) and \( Y \) are transitions in the graph. The transition traces may differ, due to edge \( E \), only when \( Y \) is originally enabled but disabled after the edge \( E \) is added. There are two possible situations. First, \( Y \) is disabled at some time and remains disabled forever, which is deadlock. The second situation is that \( Y \) is disabled at some time and will be enabled after some finite transitions. For the latter, we claim that the delay in the enabling of transition \( X \) causes no difference in system behavior. As mentioned in the definition of STG in Section 2, a transition may, but not necessarily, be triggered if it is enabled. That is, a transition may be enabled but not triggered for any arbitrary time, which behaves the same as the delay of enabling. Therefore, if adding an edge does not cause deadlock, the new STG is conservative to the original STG. □

We elaborate the TP for each behavior model in the following subsections.

**Figure 4:** STG examples for direct synthesis: (a) Staggered, (b) Back-to-Back, (c) Sneaking, (d) Liable, and (e) Independent.

**Figure 5:** STG examples for optimization: (a) Sneaking, (b) Liable, and (c) Liable.

#### 4.1 Staggered
The fork from input signal internal signal 4.2 Back-to-Back additional edges are added in optimization for of signal Note that this optimization does not guarantee output signal to be triggered because and stable with respect to signal Hence signal signal level till transition Taking the T-element of Figure 1 for example, and the SD property. An STG example is shown in Figure 4(a). The TP for Staggered model, if \( u^* \notin \mathcal{P}_s(x^-, x^+) \), is

\[
x^* = R(u^* t^-)F(t^-)
\]

Taking the T-element of Figure 1 for example, \( x^* \), \( u^* \) and \( t^* \) correspond to \( la^+, On^+ \) and \( lr^- \), respectively.

**Lemma 5.** Given a simply-driven STG with \( u^* \in \text{En}(x^+) \), \( u \) in \( PI \), and \( x \) in \( PO \), applying TP for Staggered model guarantees the underlying output signal \( x \) to be effective, stable, and precise with respect to input signal \( u \).

**Proof.** If \( u^* \notin \mathcal{P}_s(x^-, x^+) \), then signal \( u \) must retain its signal level till transition \( x^* \) is to be triggered, and transition \( u^- \) will not occur before transition \( x^* \) is triggered. On the other hand, if \( u^* \notin \mathcal{P}_s(x^-, x^+) \), then transitions \( u^* \) and \( t^- \) must have occurred before \( x^* \) should be triggered. Note that transition \( x^- \) behaves as completion detection of transition \( t^- \). Hence signal \( x \) is precise due to Proposition 1.

**4.1.2 Optimization**

In optimizating signals with transition belonging to Staggered model, we can apply simpler TP with

\[
x^* = R(u^*)
\]

**Lemma 6.** Given a simply-driven STG with \( u^* \in \text{En}(x^+) \), \( u \) in \( PI \), and \( x \) in \( PO \), applying TP as \( x^* = R(u^*) \) for Staggered model guarantees the underlying output signal \( x \) to be effective and stable with respect to signal \( u \).

**Proof.** Signal \( u \) holds its signal level at the time when \( x^* \) is to be triggered because \( u^- \) will not be enabled before \( x^- \) has been triggered. Note that this optimization does not guarantee output signal \( x \) to be precise. Nevertheless, all TPs in Back-to-Back, Sneaking, Liable and Independent models suffice to make signal \( x \) precise as we will show in later sections. As a result, optimization for Staggered model can be applied if any of the enabling transition of signal \( x \) does not belong to Staggered model. Note that no additional edges are added in optimization for Staggered model.

**4.2 Back-to-Back**

To specify TP for Back-to-Back model, an input signal \( t \) and an output signal \( i \) should be designated. In addition, an internal signal \( B_{ux} \) is introduced in order to keep signal \( x \) stable. The fork from input signal \( u \) to output signal \( x \) and to internal signal \( B_{ux} \) is designated isochronic. Signal \( i \) can be any signal satisfying \( i^* \in \text{En}(u^*) \), while signal \( t \) is chosen such that \( x^* \in \text{En}(t^*) \), \( x^- \in \text{En}(t^-) \), and \( t^- \in \mathcal{P}_s(x^-, u^-) \). Such input signal \( t \) exists due to Lemma 1 and the SD property. Figure 4(b) shows an example. The TP of Back-to-Back model is specified by

\[
B_{ux} = R(u^*, t^-)F(u^-), \quad x^* = R(u^*, B_{ux}^+)F(B_{ux}^+), \quad i^* = R(B_{ux}^+)
\]

where \( B_{ux}^+ \) has the same sign as \( x^* \). Taking the STG of Figure 6(a) for example, \( x^* \), \( u^* \), \( t^* \), and \( i^* \) correspond to \( PO_{1}^+ \), \( PI_{1}^+ \), \( PI_{2}^+ \), and \( PO_{2}^+ \), respectively.

**Lemma 7.** Given a simply-driven STG with \( u^* \in \text{En}(x^+) \), \( u \) in \( PI \), and \( x \) in \( PO \), applying TP for Back-to-Back model guarantees the underlying output signal \( x \) to be effective, stable, and precise with respect to input signal \( u \).

**Proof.** This proof is similar to the proof of Lemma 5.

**4.3 Sneaking**

**4.3.1 Direct Synthesis**

To specify TP for Sneaking model, an input signal \( t \) and output signal \( j \) have to be designated. Signal \( j \) can be any signal if \( j^* \in \text{En}(x^+) \). Signal \( t \) is chosen such that \( x^* \in \text{En}(t^*) \), \( x^- \in \text{En}(t^-) \) and \( t^- \in \mathcal{P}_s(x^-, x^-) \). Such input signal \( t \) exists due to Lemma 2 and the SD property. We also create an internal signal \( B_{ux} \) to avoid deadlock by considering the completion detection of signal \( x \). The fork from input signal \( u \) to output signal \( x \) and to internal signal \( B_{ux} \) is designated isochronic. An example is shown in Figure 4(c). The TP for Sneaking model is specified by

\[
B_{ux} = R(u^*, t^-)F(u^-), \quad x^* = R(u^*, B_{ux}^+)F(B_{ux}^+), \quad j^* = R(B_{ux}^+)
\]

where \( B_{ux}^+ \) has the same sign as \( x^* \). Taking T-element in Figure 1 for example, \( x^* \), \( u^* \), \( t^* \), and \( j^* \) correspond to \( la^+, On^+, lr^- \), and \( Or^- \), respectively.

**Lemma 8.** Given a simply-driven STG with \( u^* \in \text{En}(x^+) \), \( u \) in \( PI \), and \( x \) in \( PO \), applying TP for Sneaking model guarantees the underlying output signal \( x \) to be effective, stable, and precise with respect to input signal \( u \).

**Proof.** This proof is similar to the proof of Lemma 5.

**4.3.2 Optimization**

To simplify TP for Sneaking model, we may search an output transition \( j^* \) such that \( j^* \in \text{En}(u^*) \) and \( u^* \) is the only fanout transition of \( j^* \). If such transition exists, we select an input transition \( t^* \) satisfying \( x^* \in \text{En}(t^*) \) and \( x^- \) is the only fanin of \( t^* \), as shown in Figure 5(a). The TP can be simplified as

\[
x^* = R(u^*)F(u^-), \quad j^* = R(t^*)
\]

Taking T-element in Figure 1 for example, \( x^* \), \( u^* \), \( t^* \), and \( j^* \) correspond to \( la^+, On^+, lr^- \), and \( Or^- \), respectively.

This optimization simplify TP by adding an additional edge to the original STG, as the red dash line shows in Figure 5(a). System concurrency is reduced since transition \( u^- \) cannot occur until transition \( t^- \) completes, making it behaves similar to Staggered model. We select transition \( j^* \) and \( t^* \) with \( u^- \) and \( x^- \) being their only fanout, respectively; to avoid deadlock and minimize concurrency reduction. However, different criteria should be studied in the future.

**Lemma 9.** Given a simply-driven STG, the above optimization for Sneaking model guarantees the modified STG to be deadlock-free and conservative to the original STG.
Proof. In the Sneaking model, $x^*$ and $u^*$ are concurrent, which implies no simple cycle contains both the two transitions due to Lemma 1. Because we choose $j'$ and $t'$ with $u^*$ and $x^*$ being their only fanout, respectively, no simple cycle contains both $j'$ and $t'$. Transition $j'$ and $t'$ are therefore concurrent. Because a simply-driven STG is deadlock-free and adding an edge between two concurrent transitions will not cause deadlock, the optimization approach guarantees the modified STG to be deadlock-free and conservative to the original one due to Lemma 4.

4.4 Liable

4.4.1 Direct Synthesis

The TP for Liable model depends on the STG structure. If $x^* \in \text{En}(u^*)$, e.g., $PI_2^+ \in \text{En}(PO_2^+)$ in Figure 6(b), then the TP is

$$x^* = R(u^*)F(u^*)$$

However, if $x^* \notin \text{En}(u^*)$, then an input signal $t$ and output signals $i$ and $j$ should be designated. Signals $i$ and $j$ can be arbitrary signals as long as $i' \in \text{En}(u^*)$ and $j' \in \text{En}(u^*)$. Signal $t$ is chosen such that $x^* \in \text{En}(t^*)$, $x^* \in \text{En}(t^*)$, and $t^* \in \mathbb{P}(x^*, u^*)$. Due to Lemma 1 and the SD property signal $t$ with the above constraints exists. Under these criteria, signal $t$ does not change its signal level until $x^*$ occurs, which can be utilized as a substitute for $u^*$. In addition, an internal signal $B_{a,t}$ is introduced to keep signal $s$ stable. The fork from input signal $u$ to output signal $x$ and to internal signal $B_{a,t}$ is designated isochronic.

Figure 4(d) shows an STG example. Taking T-element in Figure 1 as another example, $x^*$, $u^*$, $t^*$, $i^*$, and $j^*$ correspond to $O_{\text{st}}^+, O_{\text{st}}^-, O_{\text{st}}^+, I_{\text{st}}^+$, and $I_{\text{st}}^-$, respectively. The TP for Liable model, if $x^* \notin \text{En}(u^*)$, is

$$B_{a,t}^+ = R(u^*, t^*)F(u^*, t^*)$$

$$x^* = R(u^*, B_{a,t}^+)F(B_{a,t}^+)$$

$$i^* = R(B_{a,t}^-)$$

$$j^* = R(B_{a,t}^+)$$

where $B_{a,t}^+$ has the same sign as $x^*$.

Lemma 10. Given a simply-driven STG with $u^* \in \text{En}(x^*)$, $u$ in PI, and $x$ in PO, applying TP for Liable model guarantees the underlying output signal $x$ to be effective, stable, and precise with respect to input signal $u$.

Proof. This proof is similar to the proof of Lemma 5.

4.4.2 Optimization

In optimizing TPs for Liable model, two possible cases are considered. If $u^* \in \mathbb{P}(x^*, x^*)$, as the case of $O_{\text{st}}^+ + I_{\text{st}}^+$ in T-element, we may simplify TP as

$$x^* = R(u^*)F(u^*)$$

This optimization may impose restriction on system concurrency. An additional edge is added to the original STG, as the red dash line shows in Figure 5(b), in order to clearly specify the transition order between $u^*$ and $x^*$. The simplified TP guarantees signal $x$ to be precise.

The other hand, if $u^* \notin \mathbb{P}(x^*, x^*)$, an input signal $t$ should be designated. Signal $t$ is chosen such that $x^* \in \text{En}(t^*)$, $x^* \in \text{En}(t^*)$, and $t^* \in \mathbb{P}(x^*, u^*)$. An STG example is shown in Figure 5(c). Taking the STG of Figure 6(b) as another example, $x^*$, $u^*$, and $t^*$ correspond to $PO_1^+$, $PI_1^+$, and $PI_2^+$, respectively. The TP in this case can be simplified as

$$x^* = R(u^*, t^*)F(u^*, t^*)$$

The TP is simplified by adding an extra edge to the original STG, as the red dash line shows in Figure 5(c). System concurrency is reduced since transition $u^*$ cannot occur until transition $t^*$ completes. The simplified TP guarantees signal $x$ to be precise.

Lemma 11. Given a simply-driven STG, the above optimization for the Liable model guarantees the modified STG to be deadlock-free and conservative to the original STG.

Proof. This proof is similar to that of Lemma 9.

4.5 Independent

The TP for Independent model is complicated due to the fact that $u^*$ is concurrent to both $x^*$ and $x^*$. Two internal signals $B_{a,t}$ and $B_{a,t}$ are added in order to keep signal $x$ stable and avoid deadlock. The fork from input signal $u$ to output signal $x$, to internal signal $B_{a,t}$ and to internal signal $B_{a,t}$ is designated isochronic. One input signal $t$ and two output signals $i$ and $j$ need to be designated. We choose signals $i$ and $j$ with $i' \in \text{En}(u^*)$ and $j' \in \text{En}(u^*)$, and designate signal $t$ satisfying $x^* \in \text{En}(t^*)$ and $x^* \in \text{En}(t^*)$. An STG example is shown in Figure 6(e). Taking the STG of Figure 9(c) as another example, $x^*$, $u^*$, $t^*$, $i^*$, and $j^*$ correspond to $PO_1^+$, $PI_1^+$, $PI_1^+$, $PO_2^+$, and $PO_2^+$, respectively. The TP for Independent model is

$$B_{a,t}^+ = R(u^*, t^*)F(u^*)$$

$$B_{a,t}^+ = R(u^*, t^*)F(u^*)$$

$$x^* = R(B_{a,t}^+, B_{a,t}^+)F(B_{a,t}^+, (B_{a,t}^+ + u^*))$$

$$i^* = R(B_{a,t}^-)$$

$$j^* = R(B_{a,t}^+)$$

where $B_{a,t}^+$ and $B_{a,t}^+$ has the same sign as $x^*$.

Lemma 12. Given a simply-driven STG with $u^* \in \text{En}(x^*)$, $u$ in PI, and $x$ in PO, applying TP for Independent model guarantees the underlying output signal $x$ to be effective, stable, and precise with respect to input signal $u$.

Proof. This proof is similar to the proof of Lemma 5.

5. CIRCUIT SYNTHESIS

Based on the behavior models and TPs introduced in Sections 3 and 4, we propose an algorithm to derive TC for every output signal of a simply-driven STG. Furthermore we design a circuit module called Up-Down-Hold module in order to implement QDI circuits from TCs.

5.1 Synthesis Algorithm

The algorithm STGSynthesis in Figure 7 firstly categorizes every enabling transition into the five behavior models for each output signal, and then TPs are applied based on the corresponding model. Optimization can be performed by applying optimizing TPs proposed in Section 4. The algorithm returns $R_0 = \{R_b(x) | x \in PO\}$ and $F_0 = \{F_b(x) | x \in PO\}$, which are the pull-up and pull-down sets for output signals.

In lines 3 to 6, function ComputePs(G, a', b') computes the transitions in $\mathbb{P}(a', b')$, which are computable in time complexity $O(|E|)$ by traversing the STG $G$. The function Classify in line 7 categorizes the behavior model of $x^*$ and its enabling transition $u^*$, as detailed in Section 3, based on the information calculated in lines 3 to 6. With the behavior model $M_b$ we derive TCs for output signals according to TPs we derived in Section 4. If reducing concurrency is acceptable, optimization is applied in line 9; otherwise, direct synthesis is applied in line 11. Since the above computation is executed at most $|E|$ iterations, the overall complexity is $O(|E|^2)$.

Theorem 2. Given simply-driven STGs, algorithm STGSynthesis guarantees every output signal to be effective, stable, and precise if no optimization is applied.

Proof. The final TC for every output signal is derived by conjuncting all the necessary conditions through the algorithm. By Lemmas 5, 7, 8, 10, and 12, the output signals are guaranteed to be effective, stable, and precise.
Algorithm \textit{STGSynthesis}

\textbf{Input:} STG $G = (V, E)$

\textbf{Output:} Triggering conditions for output signals, $R_b$ and $F_b$

1: for each $x \in PO$ do
2: for each \( u^+ \in En(x^+) \cup En(x^-) \) do
3: $P_i^e := \text{Compute}Ps(G, x^+, x^+)$
4: $P_i^p := \text{Compute}Ps(G, x^-, x^-)$
5: $P_i^f := \text{Compute}Ps(G, x^-, x^+)$
6: $P_i^b := \text{Compute}Ps(G, x^-, x^-)$
7: $M_i := \text{Classify}(G, u^+, P_i^e, P_i^p, P_i^f, P_i^b)$
8: if reducing concurrency is acceptable then
9: $(R_b, F_b, G) := \text{Optimize}(G, u^+, M_i)$
10: else
11: $(R_b, F_b) := \text{DirectSynthesis}(G, u^+, M_i)$
12: return $R_b$ and $F_b$

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{algorithm.png}
\caption{Algorithm: Derive corresponding behavior models and triggering conditions}
\end{figure}

\textbf{Theorem 3.} Given a simply-driven STG, algorithm STGSynthesis guarantees the underlying STG of the optimized circuit to be conservative to the original STG.

\textbf{Proof.} Applying TPs for \textit{Staggered}, \textit{Back-to-Back} and \textit{Independent} models guarantees the circuits to be effective and stable according to Lemmas 5, 7 and 12, and therefore no alteration is made on the STG in such cases. As for \textit{Sneaking} and \textit{Liable} models, the modified STG is conservative due to Lemmas 9 and 11 if optimization is applied. Therefore by Definition 6 the final modified STG is conservative to the input STG.

\section{5.2 Circuit Implementation}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{UDH.png}
\caption{(a) UDH module, (b) circuit example.}
\end{figure}

The triggering conditions, $R_b(x)$ and $F_b(x)$, of a signal $x$ can be directly translated into physical circuits. We propose a circuit module called Up-Down-Hold (UDH) module, which consists of a pull-up part, a pull-down part, and a state holding buffer as shown in Figure 8(a). The pull-up and pull-down sets of TCs can be directly implemented in the pull-up and pull-down parts, respectively, in a UDH module. Consider the TC example $R_b(x) = \{u^-, v^+\}$ and $F_b(x) = \{u^+, (t^+ + s^-)\}$, where $u, v, t, s$ are input signals and $x$ is the output signal to be implemented; the synthesized circuit is shown in Figure 8(b).

\section{6. EXPERIMENTAL RESULTS}

Our proposed synthesis procedure, named \textit{Stg2Qdi}, were implemented in the C++ language. All experiments were conducted on a Linux machine with a Xeon 2.3 GHz CPU and 32 GB RAM. The benchmark STGs were partly obtained from [22], from translation of the write-back unit [24], and from translation of AMBA-AHP [25]. To assess the scalability and effectiveness of our proposed method, we further created four STG families (including \textit{sequence-N}, \textit{parallel-N}, \textit{pipeline-A-N} and \textit{pipeline-B-N}, which are parameterized on \(N\) as shown in Figure 9. STG \textit{sequence-N} specifies the protocol in which $N$ processes work sequentially, with output signal $PO_i$ and input signal $PI_i$ signifying the permission and completion detection of the $i$th operation, respectively. Similarly, STG \textit{parallel-N} specifies the protocol in which $N$ processes work concurrently.

STGs \textit{pipeline-A-N} and \textit{pipeline-B-N} were built to examine how STG structures affect circuit cost.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{pipeline.png}
\caption{STGs for experiment: (a) sequence-N, (b) parallel-N, (c) pipeline-A-N, and (d) pipeline-B-N.}
\end{figure}

Since there exists no other synthesis tool that can synthesize QDI circuits from STG specifications, we only compared \textit{Stg2Qdi} to two related, but not directly comparable, well-known asynchronous synthesis tools \textit{Balsa} [26] and \textit{Petrify} [9]. \textit{Petrify} can synthesize speed independent (SI), but not the more robust QDI, circuits from STG specifications. Note that \textit{Balsa} is designed for entire system synthesis rather than controller circuit synthesis and may not be an ideal tool for comparison.

The experimental results are shown in Table 1, where Columns 2, 3, and 4 show the numbers of input signals, output signals, and edges, respectively, of the corresponding STGs listed in Column 1, Columns 5, 7, 9 and 11 show the transistor counts of circuits synthesized by \textit{Stg2Qdi} and by \textit{Balsa}, either without or with applying the optimization techniques. Columns 6, 8, 10, 12 show the runtime (in seconds) of \textit{Stg2Qdi} and \textit{Balsa}, either without or with applying optimization techniques, Column 13 shows the transistor counts of circuits synthesized by \textit{Petrify}, and Column 14 shows the runtime of \textit{Petrify} with the dominating CSC solving time reported in the parentheses. Note that an entry of \textit{\_\_\_} in the table indicates data unavailable due to either the BDD memory overflow error (indicated as M.E.) of \textit{Petrify} or timeout (indicated as T.O.) at 8000 seconds. Note that the transistors of inverters for input signals are not counted in \textit{Stg2Qdi} and in \textit{Petrify} whereas the transistor counts for \textit{Balsa} are based on technology mapping with its own standard cell library. It should be emphasized that the three methods are not directly comparable due to their different synthesis assumptions. Nevertheless, to assess the optimality of \textit{Stg2Qdi} the benchmark STG \textit{t-element} [22] provides a good reference point: Its manual design [22] uses 14 transistors whereas \textit{Stg2Qdi} uses 16 transistors, close to the manually optimized design.

Note that the STGs named \textit{"new"} in [22] are modified from their counterparts named \textit{"conv."} (conventional) to enhance system concurrency. Moreover, STGs \textit{parallel-N} are more concurrent than STGs \textit{sequence-N}. The experimental results reveal that circuits synthesized from STGs whose underlying systems behave more concurrently might require more transistors in direct implementation, but could be even smaller after optimization. This phenomenon is expectable since our optimization techniques simplify TPs by exploiting a system’s concurrency as flexibility. Although our optimization techniques add extra edges to STGs, the gained reduction on circuit sizes can be significant. The tradeoff between circuit cost and system concurrency is an interesting issue for further study. In addition, the circuit cost difference between \textit{pipeline-A-N} and \textit{pipeline-B-N} suggests that STGs with more transitions be-
longing to Staggered, Sneaking, and Liable models might require fewer transistors to implement.

For Balsa synthesis, we firstly derived the TCs for output signals of an STG through our algorithm and then translated the TCs into Balsa codes for QDI circuit synthesis. From the results of Table 1, we observe that the circuits synthesized by STGQDI require much fewer transistors than those synthesized by Balsa. However, this comparison can be unfair due to the fact that the translation from TCs to Balsa codes is indirect and makes Balsa synthesis ineffective. When STGQDI and Petrify are compared, the transistor counts of Petrify are in general smaller with a few exceptions. However, Petrify can be time consuming due to its CSC computation and, moreover, some cases cannot even be synthesized because of the state explosion issue.

7. CONCLUSION

We have analyzed the relationships between transitions of a given valid and single-cycle STG, and defined five behavior models which completely cover all possible relations. We also have proposed the first synthesis algorithm that synthesizes simply-driven STGs into QDI circuits. Experimental results suggest that our approach is effective. Case study on AMBA-AHP protocol also suggests the applicability in real-life applications.

REFERENCES