A Realization Algorithm of Asynchronous Circuits from STG
Kuan-Jen Lin and Chen-Shang Lin
Department of Electrical Engineering
National Taiwan University

Abstract
The synthesis of asynchronous circuits from the behavior
descriptions in STG is studied in this paper. A new real-
isation algorithm is proposed to synthesise asynchronous
circuits directly from STGs and thereby to maintain the
problem size proportional to the signal number only. In
previous methods, the state diagram is involved in the syn-
thesis, which has a worst-case size exponential to the signal
number. Based on the transitive lock relation[2], our real-
isation algorithm is shown to realise one-level circuit when
the given STG is $L^2$. The simple one-level realisation
ensures that the realized circuit is hazard-free under gate
delay model without any post-realisation modification.

1 Introduction
The asynchronous circuits feature the advantages of clock-
skew-free as well as low peak power over the synchronous
ones.

Among the synthesis approaches of asynchronous cir-
cuits, the STG approach proposed by Chu[1] is especially
interesting. In [1], Chu proposes the usage of Signal Tran-
sition Graph (STG) to model asynchronous behavior and
studies the properties of STG for a speed-independent cir-
cuit. To our knowledge, this is the first successful ap-
proach to synthesise asynchronous circuits from their behavioral
descriptions such as timing waveforms and hence has gen-
erated many interesting investigations. Given an STG, the
general synthesis procedure is comprised of the following
two phases: (1) ensure that the STG is feasible for realiza-
tion; and (2) synthesise for each non-input signal explic-
minimisation. However, since the size of state diagram is
exponential with respect to the number of signals, even without consid-
ering the time complexity of Boolean minimisation. More-
over a post-synthesis procedure is required to eliminate the
potential hazards.

In this paper, we propose a new realisation algorithm to
synthesize asynchronous circuits directly from STGs and
thereby to maintain the problem size proportional to the
signal number only. Based on the transitive lock rela-
tion[2], the realisation procedure is shown to realise one-
level circuits when the given STG is $L^2$. The simple one-
level realisation ensures the realised circuits to be hazard-
free under gate delay model without any post-realisation modi-
fication.

2 Feasible STG and Lock Rela-
tion
In our system, STG is used as an input specification of
asynchronous circuits, in which the operation of a circuit
behavior is described by the precedence relations between
signal transitions of the circuits. The behaviors of STG
can be interpreted as Petri net[1]. The vertices of STG (transi-
tions) represent the transitions of signals in the cir-
cuit and the arcs (places) the precedence relations between
them. The transitions of signal a denoted by $a^+$ and $a^-$
represent the rising and falling transitions respectively. In
STG, if there is an arc from a transition $t_1$ to another tran-
sition $t_2$, we denote it by $t_1 \rightarrow t_2$. And $t_1 \Rightarrow t_2 \Rightarrow \ldots \Rightarrow t_k$, $t_1, t_2, \ldots, t_k$ are in a loop such that these transitions occur
in the order of their indexes($1, 2, \ldots, k, 1$...). For the
STG under consideration in this paper, there is no choice
behavior and every signal has exactly one rising and one
falling transitions.

From [1], an STG should possess certain properties so
that a circuit can be synthesized from it. The following
definitions are based on previous works[1,4].

Definition 1 (Liveness) A STG is live if
(1) the STG is a strongly connected graph,
(2) every simple cycle contains exactly one token,
(3) transitions $t$ and $\bar{t}$ are ordered, i.e., $t \Rightarrow \bar{t}$.

Definition 2 (Persistency) A signal transition is persistent if once the transition is
enabled only the firing of the transition can disable it.

An STG is persistent if every transition in the STG is
persistent.
Definition 3 (Unique-Encoding)
An STG has a unique encoding if two different states do not have the same binary codes.

The implications of these properties can be found in [1].

From the above discussions, we say that an STG is feasible if it is live, persistent and unique-encoding.

In an STG, special relations between signals have significant impact on the feasibility. These relations are described as follows:

Definition 4 (Lock)
There is a lock relation between signals a and b, denoted as \(a \Leftrightarrow b\), iff

\[ a \rightarrow b \text{ or } b \rightarrow a \]

Note that the two signals of well-known asynchronous four-phase handshaking protocol satisfy the lock relation.

Since the four phase handshaking circuit is certainly realizable, it leads [4] to propose that lock relation is the primary condition for feasibility of STG. It is found that for feasibility, lock relation can be relaxed.

Definition 5 (Transitive Lock)
There is an \(n\)-th order transitive lock relation between signals a and b, denoted as \(L^n\) b, iff

\[ a \Leftrightarrow b \Rightarrow \exists c \in STG \text{ such that } a \Leftrightarrow c \Leftrightarrow b \]

where \(c \Leftrightarrow b\) = \(c \Leftrightarrow b\).

And there is a transitive lock relation between signals a and b, denoted as \(L^0\) b, if there exists another signal \(c\) in STG such that \(a \Leftrightarrow c \Leftrightarrow b\).

Two important results have been obtained regarding the feasibility and the lock relations[4][2].

1. A live STG has a unique encoding if there is \(L^1\) relation between any two signals.
2. A live STG is persistent and has a unique encoding, if any two signals a and b in STG with adjacent transitions, \(t_a \rightarrow t_b\) or \(t_b \rightarrow t_a\), have at least \(L^2\) relation.

The persistency and unique-encoding have been treated separately in previous works on STG approach. Generally the satisfaction of one property does not imply the possession of another. The above result allows us to explore a global solution for both persistency and unique-encoding.

3 Permissible Cubes

In realization from a feasible STG without hazard, it is desirable to produce a "clean" waveform with at most one rising and one falling transition for a cycle of STG. The signal produced from a permissible cube has this property.

A permissible cube is the minimum single-cube cover of states which are connected in the corresponding state diagram. For the STG shown in Fig. 1, both \(z\) and \(yz\) are permissible cubes while \(\bar{y} x\) is not because \(x \bar{y} z(101)\) and \(\bar{x} \bar{y} s(001)\) are not connected. The permissible cube will be used extensively in our realization. A (pseudo) cycle in the following discussion is defined as a path from any state in a state diagram such that the length of the path is just sufficient for each signal in a cube to be fired twice.

![Figure 1: An STG example and (b) its corresponding state graph with encoding by \(xyz\).](attachment:figure_1.png)

The simple but useful properties of permissible cubes are stated in the following lemmas.

**Lemma 1**: Given a cube \(z_1z_2...z_n\), then the cube is permissible for a given STG if when starting from a state in the cube, the cube changes values either once or twice for any pseudo cycle of STG.

**Lemma 2**: The nonempty intersection of two permissible cubes is also a permissible cube.

The following results will be useful to identify the permissible cubes on STG.

**Theorem 1**: A cube \(z_1z_2...z_n\) is permissible iff for any proper subset \(T\) of \(\{z_1, z_2, ..., z_n\}\), \(\exists x \in T\) and \(y \in \{z_1, z_2, ..., z_n\} - T\) such that starting from any state in the cube, \(y\) is fired before \(z+\).

Proof (only if part) Suppose there is a \(T\) such that there all \(z-\)'s fired before all \(y-\). Since the corresponding \(z-\)'s fires before \(z+\)'s, the cube is reset and then set before any \(y-\) occurs. The cube is thus bound to be reset again by these \(y-\)'s before a cycle is completed. Therefore by Lemma 1, the cube can not be permissible, which is a contradiction.

**Lemma 3**: Suppose that the cube is not permissible. Then from Lemma 1, there exists a cycle such that the cube either remains true or changes values more than twice. The former can not be true because the cube will be reset by the transitions such as \(z_{i-1}, i = 1, 2, ..., n\). For the latter case, there must exist a \(T\) such that only those in \(T\) fire twice while those in \(T\) remain the same. This is also a contradiction and the theorem is proved.

Q.E.D.

Given a cube and a state in the cube, the complexity of checking its permisibility is \(O(n^3)\) by using a preprocessed Loop Relation Table[2], where \(n\) is the number of literals in the cube.

The transitive-lock relations among signals are closely related to the construction of a permissible cube as shown below.

**Corollary 1-1**: If \(z_1 \Leftrightarrow z_2\), then the four cubes: \((z_1, \bar{z}_2, \bar{z}_1, \bar{z}_2)\) all are permissible.

**Corollary 1-2**: Let \(z_1 L y \Leftrightarrow y L z_2\) and let the two transitions of \(z_1\) and \(z_2\) between \(y+\) and \(y-\), be \(z_1+\) and \(z_2+\). Then the cubes \(z_1z_2\) and \(\bar{z}_1 \bar{z}_2\) are permissible.
4 Realization Algorithm

This section presents a procedure to realise circuits for the non-input signals in a feasible STG with transitive lock relation between each pair of signals. These circuits will be shown to be hazard-free without post-synthesis modification under gate-delay model in the next section. The realisation procedure is carried out wholly on the STG domain. Our synthesis procedure exploits extensively the permissible cubes and the transitive lock relations among signals.

The realisation is based on the following lemmas. These lemmas are originated from the waveform templates [2] which can be easily realised without hazards.

Let the immediate predecessors of \( f+ \) be \( s_1, s_2, ..., s_m \) and of \( f- \) be \( t_1, t_2, ..., t_n \). Let existing a cube \( S_j(R_j) \) covered by \( s_1s_2...s_m(t_1t_2...t_n) \) and each signal in \( S_j(R_j) \) having \( L^* \) relation with \( f+(f-) \). The relation \( \tau \) is defined as the relation of either \( x+ \Rightarrow f+ \Rightarrow x- \) or \( x- \Rightarrow f+ \Rightarrow x+ \). A persistent transition implies that it has \( L^* \) relation with all its immediate predecessors. Since the polarity of a signal can be inversed without altering the signals, we always assume \( x- \Rightarrow f+ \Rightarrow x- \) for \( \tau \).

We have three fundamental templates from the functional relations between \( S_j \) and \( R_j \).

Lemma 3: If (1) both \( S_j \) and \( R_j \) are permissible-cubes, (2) \( S_j+ \Rightarrow f+ \Rightarrow f- \) and \( R_j- \Rightarrow f- \Rightarrow R_j- \Rightarrow f+ \), then \( F=S_j+R_j \).

Lemma 4: If (1) both \( S_j \) and \( R_j \) are permissible-cubes, (2) \( S_j+ \Rightarrow R_j+ \Rightarrow S_j- \Rightarrow R_j- \) and \( S_j \Rightarrow f- \Rightarrow f+ \), then \( F=S_j+R_j \).

Note that the realization of Lemma 4 is only a combinational circuit. However, the combinational realization is not always possible. Hence we anchor our realization procedure on Lemma 3, which requires only one-level combinational circuits plus one latch or C-element. Next the realization procedure will be described.

Realization Algorithm:

Given a feasible STG with \( L^{12} \) relation, let \( f \) be an non-input signal and the immediate predecessors of \( f+(f-) \) be \( s_1, s_2, ..., s_m \) and \( t_1, t_2, ..., t_n \).

1. Foreach \( s_k, k=1, 2, ..., m \),
   - Select a shortest path from \( s_k \) to \( f \) on the lock graph.
   - Let this path be \( s_k = s_{n_0} s_{n_1} ... s_{n_q} \) (\( s_{n_0} = s_k, s_{n_q} = f \)).
   - \( F_{s_k} = s_{n_0} s_{n_1} ... s_{n_q} \).
   - Forloop \( i=0 \) to \( q \),
     - remove literal \( s_i \) from \( F_{s_k} \) unless \( s_i \) has \( L^* f+ \).
     - \( i=i+1; \)
   - Endforloop
2. Repeat Step 1 with \( f- \) replacing \( f+ \) and \( f+ \) replacing \( f- \).
3. Return \( F=F_{s_1}... F_{s_m} + F(F_{t_1}... F_{t_n}) \).

The correctness of the above procedure is established in the following theorem. Theorem 2: If each immediate predecessor of \( f+ \) and \( f- \) has at least \( L^{12} \) relation with \( f \), then the sub-circuit realizing the \( s(f) \) (reset) signal to the latch of \( f \) can be implemented with one-level circuit and the literal count of combinational part for \( f \) has the upper bound: \( 3n + 3m \), where \( n, m \) are the numbers of immediate predecessors of \( f+ \) and \( f- \) in STG, respectively.

Proof: We will show that each derived \( F_{s_k} \) (\( F_{s_k} \)) is a permissible cube and has the relation: \( F_{s_k}+ + (F_{s_k}-) \Rightarrow f+ \Rightarrow f- \Rightarrow f_2 \Rightarrow (F_{s_k}-) \Rightarrow f- \). Then \( F_{s_1}... F_{s_m} (F_{t_1}... F_{t_n}) \), the intersection of permissible cubes, is also a permissible cube and the relation \( F_{s_1}... F_{s_m} (F_{t_1}... F_{t_n}) \Rightarrow f+ \Rightarrow f- \Rightarrow f \). This result has the same literal count as reported in [4]. However our result is realized in one-level circuit with additional advantage of hazard-free.

Six cases are evaluated and the results are shown in Table 1. The first case, A/D converter, is the description of an asynchronous controller of A/D converter described in [1]. The STG is modified for \( L^{12} \) and the resultant

324
implementation needs two less literals than [1] as shown in
the first row of Table 1. However, when the maximum lock
strategy[2] is applied to satisfy feasibility for the original
STG, it needs less literals than the previous one. This
result is shown in the second row. For the other four cases,
two are from [5] [6] and last two are from [4]. From Table
1, our low complexity method obtains similar results as
those by exponential-complexity method.

The time complexity of Realisation Procedure is mainly
on the selection of the shortest path in step 1. Hence,
the complexity for the synthesis of one signal takes
\(O(N^2)\) assuming that the number of immediate predecessors
is constant, where \(N\) is the number of signals in
STG.

In the 28th DAC, a question was raised concerning the
relation between persistency and one-level single-cube re-
alisation[7]. We will answer the question in the following.

**Theorem 3:** In a live and unique-encoding STG, if a
transition of an non-input signal \(f+(f-)\) is not persistent,
then the combinational parts of direct realization for \(f\)
can not be implemented with a single-cube.

**Proof:** Let \(a\) be immediate predecessor of \(f+\), which
causes its nonpersistency. Hence \(a\) is concurrent with
\(f+\) in STG. The logic function for \(f\) is then must cover
a minterm \(m_0\) with literal \(a\), but not \(f\), because \(a\) is an
immediate predecessor, and not cover the same minterm
with \(a\) replaced by \(\overline{a}\), denoted by \(m_2\). Moreover, since \(a\)
is concurrent with \(f+\), there always exist another minterm
\(m_3\) in the cover of \(f\) containing literal \(\overline{a}\) but not \(f\). To have
single-cube realization, there must be one cube to cover
both minterms \(m_1\) and \(m_3\), which clearly can not contain
literal \(a\). Thus such cube covers \(m_3\) which belongs to the
offset of \(f\). A contradiction occurs and hence there can not
be a single-cube realization.

Q.E.D.

In other words, persistency is a necessary condition for
single-cube direct realization, which can be implemented
with one-level circuit. However, it is not sufficient
and a persistent STG may not have one-level realization.
A counter example is provided: \(a\rightarrow c+, a\rightarrow f+, \overline{c}\rightarrow \overline{d}, \overline{d}\rightarrow \overline{b}+, f\rightarrow b+, b\rightarrow \overline{e}+, e\rightarrow \overline{f}, \overline{f}\rightarrow e, \overline{e}\rightarrow \overline{c}, c\rightarrow \overline{a}, e\rightarrow \overline{b}, a\rightarrow d-, d-\rightarrow a+, \text{and } b\rightarrow \overline{a}+. \) The above STG is persistent and its
minimized realization for signal \(f\) is then \(ad + c\overline{b} + \overline{fe}, \) a
2-level realization. Moreover, it can be easily shown that
there is no 1-level solution exists.

5 **Timing Considerations**

We are to show that the circuit synthesised from the pre-
vious procedure is free of hazard under gate-delay model.
In other words, the non-zero delay is assumed to be con-
centrated in gates and the stray line delay and the delay
of simple INV gates to complement signals for the re-
alisation are negligible. Another assumption is that the
internal feedback loop delay within a latch is much smaller
than the delay on any path from the latch output to one
of its inputs.

We will first establish that the three cases considered in
Lemmas 3 and 4 can be realised in hazard-free combina-
tional circuits under gate-delay model.

Recall the conditions of static hazard for two-level
single-output combinational circuit permitting multiple in-
put changes[8]: (1) all the effects of one change reach the
output before any effects of the next change reach the
output, (2) free of Function Hazard and Logic Hazard.
Condition (1) can be met by treating those possible input
changes between two successive output changes in STG as
intermediate multiple input changes. These intermediate
changes should not affect the output values functionally
since the derived logic function has ensured that. How-
ever, under gate model, the circuit could still has different
input output value momentarily. We will justify that these
haz-
ards will not occur for the AND-OR case. The other cases
can be similarly justified. (1) An AND-to-OR circuit
is free of 0-hazards because each \(AND\) gate implementing
a permissible cube has hazard-free output. (2) An AND-
to-OR circuit is free of 1-hazards because all possible in-
put intermediate-vectors are covered by an \(AND\) gate or
the controlling-1 to \(OR\) gate is always transferred from an
slower \(AND\)-gate to a faster single-literal cube, physically
a simple wire, during the input changes as in Lemma 4.
Moreover, since the output of each \(AND\) gate is a per-
missible cube, which implies a "clean" waveform, hence no
dynamic hazard can occur at the output.

For the case of Lemma 3, it easy to see that the hazards,
static and dynamic, can not occur at the outputs of com-
bination parts by similar reasoning. The cause of haz-
ardous output in sequential circuit designed with latches
or C-elements can be categorized into the three cases: (1)
Static hazard: this is caused by the static hazard of latch
inputs, the occurrence of which is shown to be impossible
in our realisation. (2) Dynamic hazard: this may be caused
by the either one or two input transient. The former case
can not occur because the input is free of dynamic haz-
ard. For the second case, there might be a short period of
both latch inputs being one due to different delays of \(S\)
and \(R\) signals. This can be avoided by the use of C-elements
(clean input waveform is also required).

The essential hazards emphasised in traditional asyn-
chronous sequential circuit design is mainly caused by the
different path delays between state variables and input sig-
als. Since our previous discussions are based on total
state model[1], the one-level realization ensures that the
equal path delay for the same gate and the input change al-
ways propagates before its incuring state variable changes. Thus the essential hazards are eliminated in our realisation.

In summary, we have justified the circuits realised from Realisation Procedure are hazard-free under gate-delay model.

6 Conclusion

A new realisation algorithm has been developed to synthesize asynchronous circuits from the behavior descriptions in STG. The novelty of this algorithm is that the realisation is carried out directly on STGs and thereby maintain the problem size proportional to the signal number only. In previous methods, the state diagram has been involved in the synthesis, which has a worst-case size exponential to the signal number. Based on the transitive lock relation[2], the new realisation algorithm has been shown to realize one-level circuits when the STG is L". The simple one-level realisation also ensures the realised circuit to be hazard-free under gate delay model without any post-realisation modification. A more comprehensive evaluation on larger cases is currently undertaken.

References


<table>
<thead>
<tr>
<th>Circuits</th>
<th>Ours Literals (Combination)</th>
<th>Memories</th>
<th>Other's Literals (Combination)</th>
<th>Memories</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD controller(1)</td>
<td>10</td>
<td>2</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>AD controller(2)</td>
<td>9</td>
<td>2</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Stage controller</td>
<td>13</td>
<td>2</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>FIFO controller</td>
<td>12</td>
<td>2</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Van. (a)</td>
<td>12</td>
<td>2</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Van. (b)</td>
<td>8</td>
<td>3</td>
<td>9</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1. Realisation Results.