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 realization algorithm is proposed to synthesize 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 synthesis, which has a worst-case size exponential to the signal number. Based on the transitive lock relation[2], our realization algorithm is shown to realize one-level circuit when the given STG is L^{t2} . The simple one-level realization ensures that the realized circuit is hazard-free under gate delay model without any post-realization 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 circuits, the STG approach proposed by Chu[1] is especially interesting. In [1], Chu proposes the usage of Signal Transition Graph (STG) to model asynchronous behavior and studies the properties of STG for a speed-independent circuit. To our knowledge, this is the first successful approach to synthesize asynchronous circuits from their behavioral descriptions such as timing waveforms and hence has generated 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 realization; and (2) synthesize for each non-input signal explicitly[1] or implicitly[3] from state diagrams using Boolean minimization. However, since the size of state diagram is exponential with respect to the signal number, the time complexity of the synthesis procedure is also exponential with respect to the number of signals, even without considering the time complexity of Boolean minimization. Moreover a post-synthesis procedure is required to eliminate the potential hazards.

In this paper, we propose a new realization 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 relation[2], the realization procedure is shown to realize one-

level circuits when the given STG is L^{t2} . The simple one-level realization ensures the realized circuits to be hazard-free under gate delay model without any post-realization modification.

The paper is organized as follows. The relation between STG feasibility and transitive lock will be summarised in the next section. Then the relation between transitive lock and permissible cube will be established. In Section 4, the algorithm will be described and proved. Preliminary evaluation results will also be given. Then the considerations on hazards will be discussed. And finally the conclusion will be given.

2 Feasible STG and Lock Relation

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 (*transitions*) represent the transitions of signals in the circuit 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 transition t_2 , we denote it by $t_1 \rightarrow t_2$. And $t_1 \Rightarrow t_2 \dots \Rightarrow t_k$, $t_1, t_2, \dots, t_k$ are in a loop such that these transitions occur in the order of their indexes(1, 2, ..., 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 iff

- (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 (Persistence)

A signal transition is persistent if once the transition is enabled only the firing of the transition can disable it.

An STG is persistent iff 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 $a L b$, iff

$$\begin{aligned} a+ \Rightarrow b+ \Rightarrow a- \Rightarrow b- \\ \vee a+ \Rightarrow b- \Rightarrow a- \Rightarrow b+ \end{aligned}$$

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 realisable, 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 $a L^n b$

iff there exists another signal c in STG such that

$$\begin{aligned} a L c \wedge c L^{(n-1)} b \\ \text{where } c L^0 b = c L b. \end{aligned}$$

And there is a transitive lock relation between signals a and b , denoted as $a L^t b$, iff there exists an n such that a and b have an n -th transitive lock relation.

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^t 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^{t1} 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 transitions 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 x and yz are permissible cubes while $\bar{y}z$ is not because $x\bar{y}z(101)$ and $x\bar{y}z(001)$ 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: (a) An STG example and (b) its corresponding state graph with encoding by xyz .

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

Lemma 1: Given a cube $x_1x_2\dots x_n$, then the cube is permissible for a given STG iff 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 $x_1x_2\dots x_n$ is permissible iff for any proper subset T of $\{x_i \mid i=1, 2, \dots, n\}$, $\exists x \in T$ and $y \in \{x_i \mid i=1, 2, \dots, n\} - T$ such that starting from any state in the cube, $y-$ is fired before $x+$.

Proof:(only if part) Suppose there is a T such that there all $x+$'s fired before all $y-$. Since the corresponding $x-$'s fires before $x+$'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.

(if part) 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 x_i- , $i = 1, 2, \dots, n$. For the latter case, there must exist a T such that only those in T fire twice while those in $\bar{T}$ 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 permissibility 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 $x_1 L x_2$, then the four cubes: $(x_1, \bar{x}_1) \times (x_2, \bar{x}_2)$ all are permissible.

Corollary 1-2: Let $x_1 L y$ and $y L x_2$ and let the two transitions of x_1 and x_2 between $y+$ and $y-$ be x_1+ and x_2+ ($\bar{x}_1+$, $\bar{x}_2+$). Then the cubes x_1x_2 and $\bar{x}_1\bar{x}_2$ ($\bar{x}_1x_2$ and $x_1\bar{x}_2$) are permissible.

4 Realization Algorithm

This section presents a procedure to realize 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 realization 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 realization is based on the following lemmas. These lemmas are originated from the waveform templates [2] which can be easily realized without hazards.

Let the immediate predecessors of $f+$ be $s_1+, s_2+, \dots, s_n+$ and of $f-$ be $r_1+, r_2+, \dots, r_m+$. Let existing a cube $S_f(R_f)$ covered by $s_1s_2\dots s_n(r_1r_2\dots r_m)$ and each signal in $S_f(R_f)$ having L^s relation with $f+(f-)$. The relation $x L^s f+$ 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^s relation with all its immediate predecessors. Since the polarity of a signal can be inversed without altering the discussions, we always assume $x+ \Rightarrow f+ \Rightarrow x-$ for $x L^s f+$ in the sequel. We have three fundamental templates from the functional relations between S_f and R_f .
Lemma 3: If (1) both S_f and R_f are permissible-cubes, (2) $S_f+ \Rightarrow f+ \Rightarrow S_f- \Rightarrow f-$ and $R_f+ \Rightarrow f- \Rightarrow R_f- \Rightarrow f+$, then $F = S_f + F\overline{R_f}$.

Lemma 4: If (1) both S_f and R_f are permissible-cubes, (2) $S_f+ \Rightarrow R_f+ \Rightarrow S_f- \Rightarrow R_f- (S_f+ \Rightarrow R_f- \Rightarrow S_f- \Rightarrow R_f+)$, then $F = S_f\overline{R_f} (F = S_f + \overline{R_f})$.

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^{t2} relation, let f be an non-input signal and the immediate predecessors of $f+(f-)$ be $s_1, \dots, s_n (r_1, \dots, r_m)$.

1. Foreach $s_k, k=1, 2, \dots, n.$ {
 Select a shortest path from s_k to f on the lock graph.
 Let this path be $s_{k0}, s_{k1}, \dots, s_{ki+1} (s_{k0} = s_k, s_{ki+1} = f)$.
 $F_{s_k} = s_{k0} s_{k1} \dots s_{ki}$.
 Forloop $i = 0$ to i , {
 remove literal s_{ki} from F_{s_k} unless $s_{ki} L^s f+$;
 $i = i + 1$;
 }endforloop
 }endforeach
2. Repeat Step 1 with $f-$ replacing $f+$ and r replacing s .
3. Return $F = F_{s_1} \dots F_{s_n} + F(\overline{F_{r_1}} \dots \overline{F_{r_m}})$.

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^{t2} relation with f , then the sub-circuit realizing the *set(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 $F_{s_k} (F_{r_k})$ is a permissible cube and has the relation: $F_{s_k} + (F_{r_k}-) \Rightarrow f+ \Rightarrow F_{s_k} - (F_{r_k}+) \Rightarrow f-$. Then $F_{s_1} \dots F_{s_n} (F_{r_1} \dots F_{r_m})$, the intersection of permissible cubes, is also a permissible cube and the relation $F_{s_1} \dots F_{s_n} + (F_{r_1} \dots F_{r_m}-) \Rightarrow f+ \Rightarrow F_{s_1} \dots F_{s_n} - (F_{r_1} \dots F_{r_m}+) \Rightarrow f-$ is also preserved. Furthermore, since $F_{s_1} \dots F_{s_n} (F_{r_1} \dots F_{r_m})$ contains all the immediate predecessors of $f+(f-)$, then by Lemma 3, we have $F_{s_1} \dots F_{s_n} + F(\overline{F_{r_1}} \dots \overline{F_{r_m}})$ which has a one-level realization with a latch.

Now we will show that F_{s_k} is indeed a permissible cube and has $F_{s_k}+ \Rightarrow f+ \Rightarrow F_{s_k}- \Rightarrow f-$. F_{r_k} can also be shown in a similar way.

Since the given STG possesses L^{t2} relation and persistency, it is always true that $s_{k0} L^s f+$ and $s_{ki} L^s f+$ with $i \leq 2$. Two cases are to be considered.

(1) All $s_{k0} \dots s_{ki}$ have L^s relation with $f+$. Then the cube $F_{s_k} = s_{k0} \dots s_{ki}$ is permissible and the cube is on when $f+$ occurs.

(2) The only signal may not have L^s relation with $f+$ is s_{ki} . Assume that L^s does not exist between s_{ki} and $f+$. Then because of the lock relations $s_{k0} L s_{ki} L s_{ki+1}$, s_{k0} and s_{ki+1} , must have same transition phase between the two transitions of s_{ki} , otherwise it can be shown that a contradiction of $s_{ki} L s_{ki+1}$ will occur. Hence $F_{s_k} = s_{k0} s_{ki+1}$ is permissible from Corollary 1-2.

The signals in F_{s_k} from both cases all have L^s relation with $f+$. Hence, it set on before $f+$ and remains on until $f+$ having fired. Moreover, since it contains s_{ki} so that the cube will be set off before $f-$ and stay off until $f-$ is fired. Thus $F_{s_k}+ \Rightarrow f+ \Rightarrow F_{s_k}- \Rightarrow f-$.

The remaining part is the upper bound of literals. This is easy to see since there are at most three literals for each immediate predecessor of $f+$ and $f-$. The upper bound is then $3n + 3m$ and hence the theorem is proved. Q. E. D.

To illustrate the Procedure, the signal Lr shown in Fig. 2(a) [4] is synthesized. Firstly, the immediate predecessors of $Lr+$ are $Za-$ and $Da-$. From the corresponding lock graph in Fig. 2(b), the shortest path from Za to Lr is found to be $Za - Zr - Lr$, hence $F_{Za} = \overline{Za} \overline{Zr}$. Similarly, $F_{Da} = \overline{Da} \overline{Dr}$. For $Lr-$, we have $F_{Zr} = Zr$ and $F_{Dr} = Dr$. Therefore, the function obtained from step 3 is $F = F_{Za} F_{Da} + F_{Zr} F_{Dr} = \overline{Za} \overline{Zr} \overline{Da} \overline{Dr} + F_{Zr} F_{Dr}$. 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^{t2} and the resultant

Figure 2: (a) An STG example and (b) its corresponding lock graph.

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 Realization 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 realization[7]. We will answer the question in the following. **Theorem 3:** In a live and unique-encoding STG, if a transition of a non-input signal $f+(f-)$ is not persistent, then the combinational parts of direct realization for $f(\bar{f})$ 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_1 with literal a , but not f , because $a+$ is an immediate predecessor, and not cover the same minterm with a replaced by $\bar{a}$, denoted by m_2 . Moreover, since $a-$ is concurrent with $f+$, there always exists another minterm m_3 in the cover of f containing literal $\bar{a}$ 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_2 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: $a+ \rightarrow c+$, $a+ \rightarrow f+$, $c+ \rightarrow d+$, $d+ \rightarrow b+$, $f+ \rightarrow b+$, $b+ \rightarrow e+$, $e+ \rightarrow f-$, $f- \rightarrow e-$, $e- \rightarrow c-$, $c- \rightarrow a-$, $c- \rightarrow b-$, $a- \rightarrow d-$, $d- \rightarrow a+$, and $b- \rightarrow a+$. The above STG is persistent and its minimized realization for signal f is then $a\bar{d} + c\bar{b} + f\bar{e}$, 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 synthesized from the previous procedure is free of hazard under gate-delay model. In other words, the non-zero delay is assumed to be concentrated in gates and the stray line delay and the delay of simple *INV* gates to complement signals for the realization 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 realized in hazard-free combinational circuits under gate-delay model.

Recall the conditions of static hazard for two-level single-output combinational circuit permitting multiple input 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. However, under gate model, the circuit could still has different output value momentarily. We will justify that these hazards 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 input 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 permissible 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 combinational parts by similar reasoning. The cause of hazardous output in sequential circuit designed by 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 realization. (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 hazard. 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 emphasized in traditional asynchronous sequential circuit design is mainly caused by the different path delays between state variables and input signals. 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 incurring state variable changes. Thus the essential hazards are eliminated in our realization.

In summary, we have justified the circuits realized 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^2 . The simple one-level realisation also ensures the realised circuit to be hazard-free under gate delay model without any post-realization modification. A more comprehensive evaluation on larger cases is currently undertaken.

References

- [1] T. A. Chu, "Synthesis of Self-Timed Control Circuits from Graphs: An Example," *Proceedings of the International Conference on Computer Design, IEEE*, pp. 565-571, 1986.

- [2] K. J. Lin and C. S. Lin, "Automatic Synthesis of Asynchronous Circuits," *Proceeding of 28th Design Automatic Conference*, pp. 296-301, 1991.
- [3] L. Lavagno, K. Keutzer, A. Sangiovanni-Vincentelli, "Algorithms for Synthesis of Hazard-free Asynchronous Circuits," *Proceeding of 28th Design Automatic Conference*, pp. 302-308, 1991.
- [4] Peter Vanbekbergen, Francky Catthoor, Jef Van Meerbergen, Hugo DE Man, "Optimised Synthesis of Asynchronous Control Circuits from Graph-theoretic Specifications," *Proceeding of the International Conference on Computer-Aided Design*, pp. 184-187, 1990.
- [5] T. A. Chu, "Design of VLSI Asynchronous FIFO Queues for Packet Communication Networks," *Proceedings of the International Conference on Parallel Processing, IEEE*, pp. 397-400, 1986.
- [6] T. A. Chu, "A Design Methodology for Concurrent VLSI System," *Proceedings of the International Conference on Computer Design, IEEE*, pp. 407-410, 1985.
- [7] Peter A. Beerel, "Private Communication".
- [8] S. H. Unger, *Asynchronous Sequential Switching Circuits*, Wiley Interscience, 1969.

Circuits	Ours		Other's		
	Literals (Combinational)	Memories	Literals (Combinational)	Memories	
AD controller(1)	10	2	12	2	[1]
AD controller(2)	9	2	12	2	[1]
Stage controller	13	2	13	2	[6]
FIFO controller	12	2	12	2	[5]
Van. (a)	12	2	12	2	[4]
Van. (b)	8	3	9	2	[4]

Table 1. Realization Results.