Hazard-Free Synthesis and Decomposition of Asynchronous Circuits

Ren-Der Chen, Jer Min Jou, and Yeu-Horng Shiau

Department of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.

Abstract

In this paper, we solve the problems of hazard-free synthesis and decomposition of asynchronous speed-independent circuits for technology mapping. All high fanin gates are decomposed into gates that can be implemented by the gate library. We first analyze the conditions where hazards may occur during decomposition and then give corresponding strategies to solve them. All the proposed algorithms have been implemented and applied to the asynchronous benchmarks to verify their correctness. Experimental results show that less area is required in our final implementations.

1 Introduction

Asynchronous speed-independent circuits are hazard-free under the unbounded gate delay assumption that gate delays are unbounded but finite, and there is no delay on wires. Several works have been done on the implementation of speed-independent circuits using basic gates [1, 3, 6]. Gates used in their implementations are assumed to have no fanin limits. Decomposition of speed-independent circuits has also been done by [2, 5, 10]. The method in [10] decomposes high fanin gates hazard-freely only for some gates since further implementation space searching and gate sharing are not considered. In [2], the sequential element in a speed-independent circuit is decomposed into two sequential elements, or a sequential and a combinational element after the correctness conditions are analyzed. In [5], they perform both combinational and sequential decompositions of complex gates while preserving the original behavior and speed-independence.

Specifying and analyzing speed-independent circuits based on the *Signal Transition Graph (STG)* has the advantage that its problem complexity is just proportional to the number of signals. Early synthesis of asynchronous circuits on the STG level has been done in [4, 8, 9]. Here, also based on the STG level, we analyze the conditions where hazards may occur while decomposing a speed-independent circuit into composed of lower-fanin gates implementable by the gate library. Some strategies for overcoming these problems are also proposed. Then the proposed alogrithms are implemented and incorporated into our earlier asynchronous synthesis system [4] for the hazard-free synthesis and decomposition of speed-independent circuits.

2 Preliminaries

We first introduce the basic properties inherent in our input specification, i.e., signal transition graph (STG).

2.1 Petri net

A *Petri net* [7] is a quadruple $N = (P, T, F, m_0)$, where P is a set of places, T is a set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relations between places and transitions, and m_0 is the initial marking. A transition is *enabled* when all its fanin places are marked. When an enabled transition *fires*, it removes a token from each fanin place and adds a token to each fanout place. A *simple path* in a Petri net is a sequence of nodes where any two adjacent nodes satisfy the flow relation and no node appears on it more than once. A *simple cycle* is a simple path besides that the first and last nodes are identical.

2.2 Signal transition graph

A *signal transition graph (STG)* is an interpreted Petri net whose transitions are interpreted as the rising or falling transitions of signals. For a signal a with multiple transitions, a^*_i is used to specify the i-th generic transition. A transition b^*_i is a *trigger* transition of a^*_i if it's a fanin transition of a^*_i. And it's *persistent* with respect to a^*_i if its next transition b^*_{i+1} is ordered with a^*_i. An STG can be decomposed into a set of *marked graphs (MGs)* where each place has only one fanin and one fanout transition. Two transitions are *ordered* if there exists a simple cycle to which they belong, and are *concurrent* if they belong to the same MG but not ordered, otherwise they are in *conflict*.

3 Target circuit architecture and implementation requirements

3.1 Target circuit architecture

In Fig. 1(a), the architecture of a *signal network* for implementing signal a is shown. It contains the combinational parts, *set* and *reset transition networks*, and a storage unit, *C-element*. The behavior of a C-element, with two inputs A, B and one output C, can be decribed as $C = A*B + (A+B)*C$. A gate is *enabled (disabled)* if its output is ready for a rising (falling) transition. The set and reset transition networks

are responsible for enabling and disabling the C-element respectively. The transition network, before decomposition, is composed of only one AND gate with possible high-fanin inputs. Our purpose here is to decompose it into cascaded gates implemented by gates from the gate library, as shown in Fig. 1(b).

3.2 Speed-independent and hazard-free implementation requirements

In a speed-independent circuit, due to the lack of a global clock, no internal or output signal transition is allowed to occur unexpectedly. Each output transition should follow the input STG specification, whereas the internal transitions, though not perceived by the outside environment, should be *acknowledged* by other perceivable output signals to guarantee its completion. The conditions for speed-independent circuits to be implemented using unlimited fanin AND gates, called *monotonous cover* (MC) conditions, have been given in [2, 6].

Definition 1 *A gate changes monotonously if it experiences a rising and falling transition at most once after a tracing of the STG from the initial marking.*

Theorem 1 *For a signal network to work speed-independently without hazards, the following conditions have to be satisfied: (1) each transition network is enabled only after all the trigger transitions have been fired, and should be disabled before the next transition is enabled, (2) no two transition networks can be enabled at the same time, and (3) each transition network should change monotonously.*

Due to the *one-hot* property mentioned in condition (2), any decomposition of the OR gate is still hazard-free. Hence the OR gate can be decomposed as necessary without any additional condsiderations. Here, we only pay our attention to the decomposition of high-fanin AND gates while preserving the inherent hazard-free properties at the same time.

4 Requirements for hazard-free decomposition

For two cascaded gates, here the gates refer to AND gates if not particularly specified, the rising transition on the intermediate gate can be acknowledged by the rising transition on the final gate. Since the rising transition on the final gate is caused by the intermediate gate *and* the primary inputs. However, such acknowledgement property can not always be applied to the intermediate *falling* transition. Since the falling transition on the final gate can be caused either by the intermediate gate *or* by the primary inputs.

Besides, since the transition on the final gate changes monotonously, the high-fanin gate has to be decomposed so that the intermediate gate also changes monotonously to solve the acknowledgement problem of the intermediate falling transition. Here some strategies are proposed for this.

4.1 Direct acknowledgement

Since the transition network of a^*_i should be disabled before a^*_{i+1} is enabled, this disabled condition may be caused by the falling transition either on the intermediate gate or on the primary inputs of the final gate. The intermediate falling transition can hence be acknowledged by the final gate if the transition network can be guaranteed to be disabled by the intermediate gate but not by any one of the primary inputs of the final gate.

Definition 2 *A primary input z of the final gate is acknowledgement-free if its next transition z^*_{i+1} is enabled after transition a^*_{i+1} is fired.*

Theorem 2 *For two cascaded AND gates in a transition network, if all the primary inputs of the final gate are acknowledge-free, both the rising and falling transitions on the intermediate gate can be acknowledged by the final gate.*

For example, a 3-input gate is decomposed into two cascaded 2-input gates in Fig. 2(a). If z- fires after a- as shown in Fig. 2(b), no hazard occurs since the final falling transition is caused by the intermediate gate. But in Fig. 2(c), the final falling transition can be either caused by y- or z-.

4.2 Changing the primary inputs of the final gate

When synthesizing the transition network of transition a^*_i, the signals necessary for a monotonous cover can be properly rearranged so that the intermediate gate still keeps changing monotonously while making all the primary inputs of the final gate acknowledgement-free. This, however, will increase the synthesis time because the searching space is enlarged.

4.3 New acknowledgement wire

When the primary inputs of the final gate can not be made acknowledgement-free after rearranging the signals, the intermediate falling transition should be acknowledged elsewhere. One strategy proposed in [10] solves this by inserting an acknowledgement wire which connects this intermediate gate, with an inverter attached, to one or more gates in the complementary transition network. However, this works only when the gate is still implementable after the additional acknowledgement wire is inserted.

4.4 OR gate substitution

This strategy replaces the non-acknowledgement-free signal with another acknowledgement-free one. The signal is inserted such that only an OR gate is enough for its implementation. To avoid the additional area overhead by inserting new signals, the basic criteria followed here is not to introduce additional trigger transitions to other signals. Take Fig. 2(a) for example, the rules for inserting a new acknowledgement-free signal i in the new STG can be simply described as follows:

1. The fanin transition of $i+$ is $z+$,
2. The fanout transition of $i+$ is $a+$,
3. The fanin transitions of $i-$ are $z-$ and $a-$, and
4. The fanout transitions of $i-$ can be those existing on the paths from $i-$ to $z+$.

The *ebergen.g* circuit shown in Fig. 3 is used as another example for decomposition. Fig. 4(a) is the resulting STG by inserting a new signal i for OR gate substitution. Here the fanin transitions of $i-$ are supposed to be $a+/1$ and $x-$, and the arc from $x-$ to $i-$ can be removed due to its redundancy. Fig. 4(b) is its syntehsized result.

4.5 AND gate substitution

Here an additional signal i is inserted such that it reflects the behavior of the intermediate gate. Also take Fig. 2(a) as an example, to make i a signal implemented by AND gate, the rules for inserting transitions $i+$ and $i-$ in the new STG can also be simply described as follows:

1. The fanin transitions of $i+$ are $x+$ and $y+$,
2. The fanout transition of $i+$ is $a+$,
3. The fanin transition of $i-$ is either $x-$ or $y-$ but not both, and
4. The fanout transitions of $i-$ can be those existing on the paths from $i-$ to $x+$ and $y+$.

Fig. 5(a) is the resulting STG by inserting a new signal i for AND gate substitution. Here the fanin transitions of $i+$ are supposed to be $c+$ and $d+$, and the arc from $c+$ to $i+$ can be removed also due to its redundancy. Fig. 5(b) is its implementation after synthesis.

5 Experimental results

The strategies for dealing with the decomposition problems have been implemented and incorporated into our earlier synthesis system [4]. In our experiments, the maximum fanin of AND gates is assumed to be two. So every circuit is synthesized to be implemented using only 2-input AND and OR gates. After a new signal is inserted, the newly constructed STG has to be resynthesized until the resulting circuit can be implemented by gates from the gate library. All the experimental results can be obtained by our synthesis and decomposition system within a few seconds. It takes only a few minutes even for some larger examples.

The results are shown in Table 1. Columns "#Adds." and "#Iters." respectively represent the number of inserted signals and the number of iterations for resynthesis. To compare with [5], our results are also shown as #literals/#C-elements. When the storage unit, i.e., C-element, is counted as three literals, we can obtain the total number of literals used in our results and those in [5]. From the results, we can see that our synthesized results use less storage units and hence less area.

6 Conclusions

In this paper, we have investigated the problems of synthesis and decomposing an asynchronous speed-independent circuit composed of high-fanin gates. Some strategies, analyzed on the STG level, have been proposed for solving these problems. The algorithms have also been implemented and incorporated into our earlier synthesis system. All the asynchronous benchmarks can be easily synthesized using our synthesis system with gates supported by the gate library.

References

[1] P. A. Beerel and T. H.-Y. Meng. Automatic gate-level synthesis of speed-independent circuits. In *Proc. International Conference on Computer-Aided Design*, pages 581-586, 1992.

[2] S. M. Burns. General conditions for the decomposition of state holding elements. In *Proc. Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 48-47 1996.

[3] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev. Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. In *IEICE Trans. Inf. & Syst.*, vol. E80-D, no. 3, pages 315-325, March 1997.

[4] J.-M. Jou, R.-D. Chen and K.-M. Lin. An integrated synthesis system for speed-independent asynchronous circuits. In *Proc. International Symposium on Circuits and Systems*, pages 1600-1603, 1997.

[5] A. Kondratyev, J. Cortadella, M. Kishinevsky, L. Lavagno and A. Yakovlev. Technology mapping for speed-independent circuits: decomposition and resynthesis. In *Proc. Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 240-253 1997.

[6] A. Kondratyev, M. Kishinevsky, B. Lin, P. Vanbekbergen and A. Yakovlev. Basic gate implementation of speed-independent circuits. In *Proc. Design Automation Conference*, pages 56-62, 1994.

[7] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4): 541-580, April 1989.

[8] S.-B. Park and T. Nanya. Synthesis of asynchronous circuits from signal transition graph specifications. In *IEICE Trans. Inf. & Syst.*, vol. E80-D, no. 3, pages 326-335, 1997.

[9] E. Pastor, J. Cortadella, A. Kondratyev and O. Roig. Structural methods for the synthesis of speed-independent circuits. In *Proc. European Design and Test Conference*, pages 340-247, 1996.

[10] P. Siegel and G. De Micheli. Decomposition methods for library binding of speed-independent asynchronous designs. In *Proc. International Conference on Computer-Aided Design*, pages 558-565, 1994.

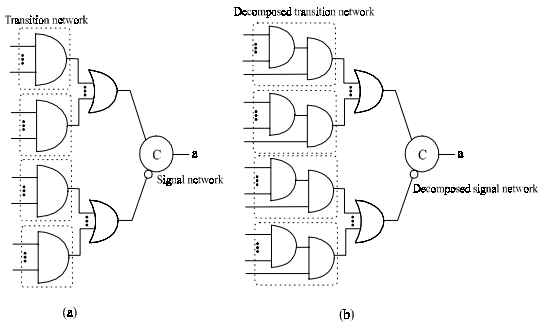

Fig. 1: Target circuit architecture: (a) No decomposition and (b) After decomposition.

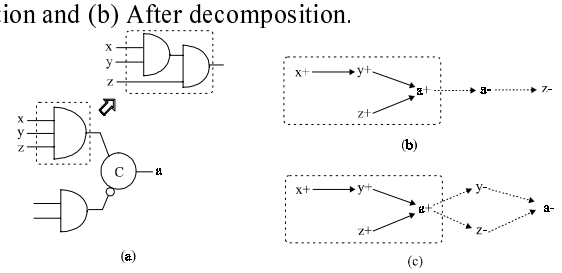

Fig. 2: A decomposed example: (a) Decomposition of a 3-input AND gate, (b) Direct acknowledgement, and (c) Additional acknowledgement required.

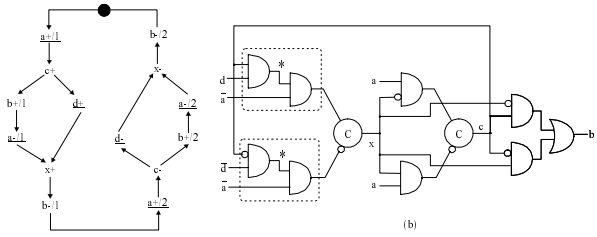

Fig. 3: Circuit example (*ebergen.g*): (a) STG and (b) Implementation.

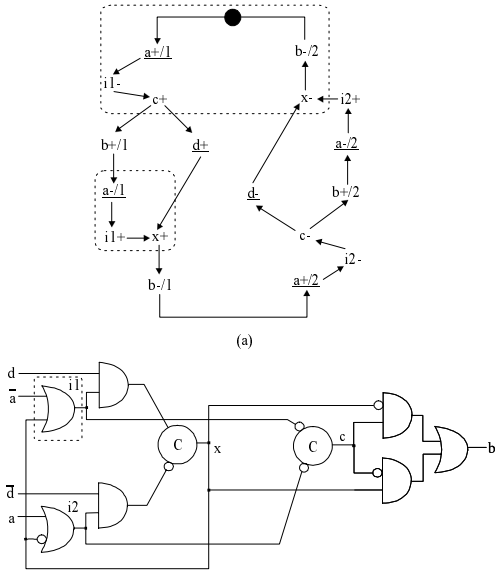

Fig. 4: Decomposition of *ebergen.g* by OR gate substitution: (a) STG and (b) Implementation.

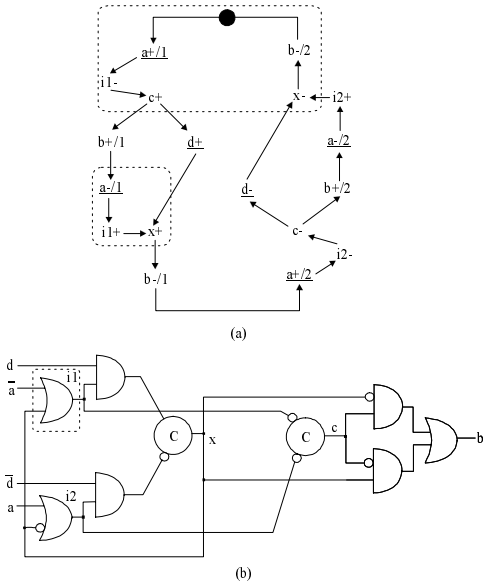

Fig. 5: Decomposition of *ebergen.g* by AND gate substitution: (a) STG and (b) Implementation.

Circuits	# Signals/ # States	#Adds.	#Iters.	Ours	[5]
alloc-outbound	9/21	-	-	16/3	15/4
chu133	7/24	-	-	14/2	13/1
chu150	6/26	-	-	13/1	17/2
converta	5/26	2	2	17/3	15/3
ebergen	5/18	2	1	14/2	23/3
hazard	4/12	2	1	10/2	14/2
master-read	14/2108	2	4	32/7	37/9
mp-forward-pkt	8/22	-	-	16/3	14/3
nak-pa	10/58	-	-	22/4	24/4
nowick	6/20	-	-	18/1	18/1
ram-read-sbuf	11/39	-	-	23/4	23/4
rcv-setup	5/14	-	-	9/1	11/1
rpdft	5/22	-	-	18/0	22/1
sbuf-ram-write	12/64	1	1	23/2	29/6
sbuf-send-ctl	8/27	1	1	19/3	27/5
sbuf-send-pkt2	9/28	2	3	30/3	30/3
vbe10b	11/256	2	4	42/7	33/7
vbe5b	6/24	1	1	13/2	13/2
vbe6a	10/192	-	-	36/6	19/7
wrdatab	10/216	5	5	54/7	54/6
				459/67	451/74
Area				660	673

Table 1: Experimental results