[10] K. Y. Yun and D. L. Dill. Unifying synchronous/asynchronous state machine synthesis. In *Proceedings IEEE 1993 ICCAD* Digest of Papers. IEEE Computer Society Press, 1993.

[11] A. J. Martin. Programming in VLSI: from communicating processes to delay-insensitive VLSI circuits. In C.A.R. Hoare, editor, *UT Year of Programming Institute on Concurrent Programming*. Addison-Wesley, 1990.

[12] A. Kondratyev, M. Kishinevsky, B. Lin, P. Vanbekbergen, and A. Yakovlev. Basic gate implementation of speed-independent circuits. In *Proc. ACM/IEEE Design Automation Conference*, 1994.

[13] T. H.-Y. Meng, R. W. Brodersen, and D. G. Messerschmitt. Automatic synthesis of asynchronous circuits from high-level specifications. *IEEE Transactions on Computer-Aided Design*, 8(11):1185–1205, November 1989.

[14] H. Hulgaard and S.M. Burns. Bounded delay timing analysis of a class of CSP programs with choice. In *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 2–11, 1994.

[15] C. J. Myers and A. J. Martin. The design of an asynchronous memory management. Technical Report CS-TR-93-30, California Institute of Technology, 1993.

Table 1: Experimental results.

Example	# of Gates	# of Gates to Decompose	AND/OR/C Area (xtors)	AND/OR/C Latency (inv)	gC Library Area (xtors)	gC Library Latency (inv)	Iter (num)
tsbm (4)	15	1	122	8.1	70	4.9	1
tsbm (3)	15	1	122	8.1	70	4.9	1
tsbm (2)	15	3	154	7.2	87	5.7	25
SELopt (4)	11	0	66	5.3	45	3.8	0
SELopt (3)	11	2	70	5.3	53	3.8	1
SELopt (2)	11	4	108	8.5	67	4.2	3
MMU (4)	27	4	186	5.8	132	5.2	14

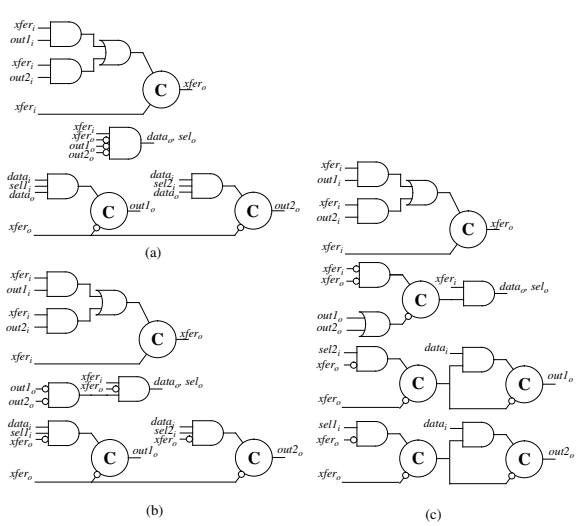

Figure 7: The gate-level timed circuit implementation of the *SELopt* example (a) before decomposition; (b) after decomposition to 3-input gates; and (c) after decomposition to 2-input gates.

Acknowledgments

We would like to thank Dr. Polly Siegel of Hewlett Packard and Professor Luciano Lavagno of Politecnico University, Torino, Italy for their advice and comments on earlier versions of this manuscript. We are also especially grateful to Professor Steve Burns of the University of Washington and Professor Alain Martin of Caltech for many stimulating discussions about timed circuits.

References

[1] C. J. Myers, T. G. Rokicki, and T. H.-Y. Meng. Automatic synthesis of gate-level timed circuits with choice. In *16th Conference on Advanced Research in VLSI*, pages 42–58. IEEE Computer Society Press, 1995.

[2] C. J. Myers and T. H.-Y. Meng. Synthesis of timed asynchronous circuits. *IEEE Transactions on VLSI Systems*, 1(2):106–119, June 1993.

[3] T. G. Rokicki and C. J. Myers. Automatic verificaton of timed circuits. In *International Conference on Computer-Aided Verification*, pages 468–480. Springer-Verlag, 1994.

[4] P. Siegel, G. DeMicheli, and D. Dill. Automatic technology mapping for generalized fundamental-mode asynchronous designs. In *Proceedings of the 30th ACM/IEEE Design Automation Conference*, 1993.

[5] P. A. Beerel and T. H.-Y. Meng. Logic transformations and observability don't cares in speed-independent circuits, 1993. In collection of papers of the *ACM International Workshop on Timing Issues in the Specification of and Synthesis of Digital Systems*.

[6] P. Siegel. *Automatic Technology Mapping for Asynchronous Designs*. PhD thesis, Stanford University, February 1995.

[7] P. A. Beerel, C. J. Myers, and T. H.-Y. Meng. Automatic synthesis of gate-level speed-independent circuits, November 1994. Submitted for publication in *IEEE Transactions on Computer-Aided Design*.

[8] L. Lavagno, K. Keutzer, and A. Sangiovanni-Vincentelli. Synthesis of hazard-free asynchronous circuits with bounded wire delays. *IEEE Transactions on Computer-Aided Design*, 14(1):61–86, January 1995.

[9] T. G. Rokicki. *Representing and Modeling Circuits*. PhD thesis, Stanford University, 1993.

4.4 Example

We present another example, an optimized version of the port selector (*SELopt*) originally described in [1], to illustrate the application of our decomposition procedure to a circuit with conditional behavior, or choice. Part of the original orbital net for the *SELopt* example is shown in Figure 6(a), and the original gate-level timed circuit implementation is shown in Figure 7(a). If we restrict our library to gates with a maximum fanin of 3, there is a 4-input AND gate that is shared to implement sel_o and $data_o$ which must be decomposed. A new signal is added for each of these signals, but we concentrate on the signal x_1 which is added to decompose sel_o. Part of the orbital net after a decomposition of sel_o is shown in Figure 6(b). This decomposition has an extracted set which consists of the context signals $\neg out1_o$ and $\neg out2_o$. The procedure detects that the corresponding transitions on these context signals occur on disjoint paths, so these transitions share a single place that is added to the preset of $x_1\uparrow$. Since there are two falling transitions on the signal sel_o, the procedure adds two falling transitions on the new signal x_1. Applying synthesis to this new specification produces the decomposed implementation shown in Figure 7(b). If we further restrict the library to only contain 2-input gates, there are three gates which must be decomposed. The resulting implementation is shown in Figure 7(c).

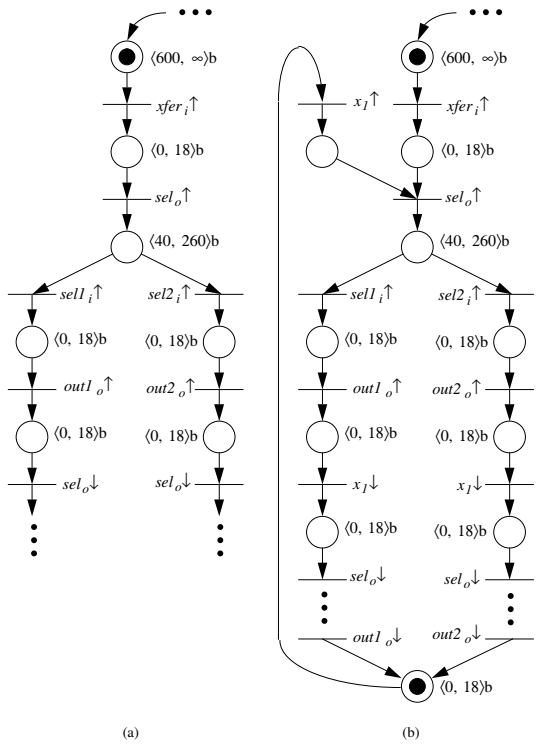

Figure 6: (a) Part of the orbital net for the *SELopt* example, and (b) part of the orbital net after a decomposition of the signal sel_o.

4.5 Results

The decomposition procedure described in this paper has been automated within the CAD tool ATACS, and it has been used to map several examples as reported in Table 1. First, a timed version of the target-send burst-mode (*tsbm*) cycle of a SCSI data transfer controller [10] is synthesized using gate libraries with a maximum fanin of 4, 3, and 2. The next three rows are implementations of the optimized port selector (*SELopt*) [1] also using libraries with a fanin of 4, 3, and 2. The last example is an asynchronous memory management unit [15].

The gate library used for each example contains gates with a maximum fanin size as specified in parentheses next to the name of the example. The next two columns give the number of gates in the standard C-implementation as well as the number of gates that are larger than the maximum fanin and must be decomposed. The area and latency for the decomposed standard C-implementation are given in the next two columns followed by the area and latency after the implementation is mapped to a library which contains generalized C-elements. Area is reported as the implementation's transistor count. Latency is the longest delay through a block of logic generating an output transition driving a fanout of 4, and it is reported normalized to the delay of a single inverter (about $300ps$ for $0.8\mu m$ CMOS process at nominal conditions). Finally, the number of iterations necessary to decompose the high-fanin gates is shown.

5 Conclusions

We have proposed an efficient procedure for the technology mapping of timed circuits to practical gate libraries. Our procedure begins by synthesizing an unlimited fanin gate-level timed circuit implementation from an orbital net specification. Next, various decompositions are applied using resynthesis to decompose all high-fanin gates. Techniques are described that make use of timing analysis to significantly reduce the number of decompositions which need to be analyzed. After decomposition, the resulting limited-fanin timed circuit implementation is mapped to the given gate library. The resulting implementations are both area efficient and deliver high performance as timing is utilized throughout the design procedure to optimize the implementation and hazard-freedom is achieved without the addition of delay elements.

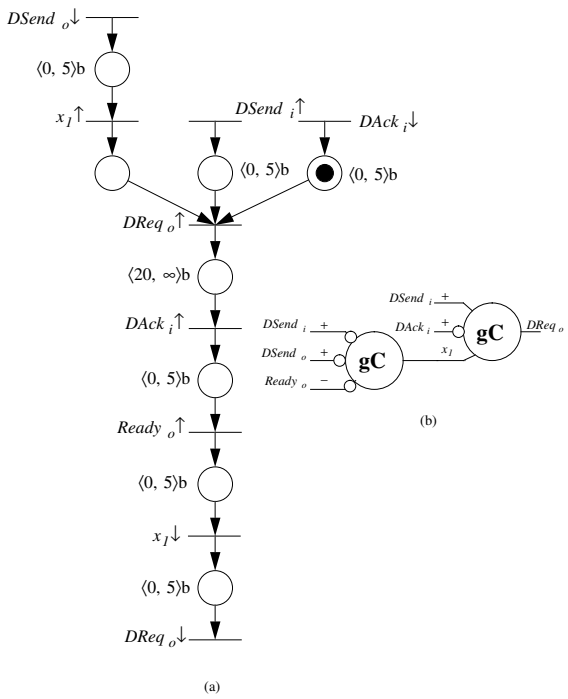

Figure 4: (a) Part of the orbital net for a decomposition using a context signal, and (b) corresponding generalized C-implementation.

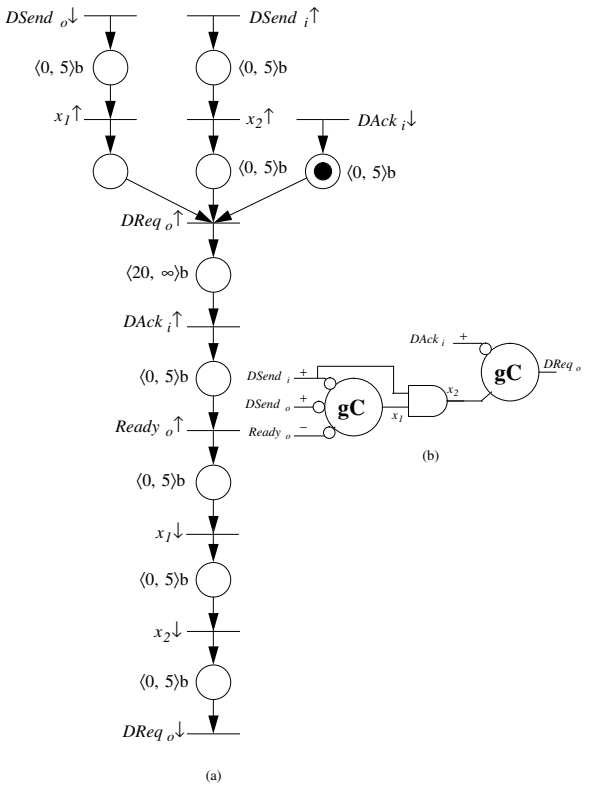

Figure 5: (a) Part of the orbital net for a multi-level decomposition, and (b) corresponding generalized C-element implementation of $DReq_o$ with a maximum fanin of two.

and select the one with the lowest cost that decomposes all high-fanin gates to gates found in the library. As a heuristic to speedup the process, our procedure exits after a decomposition is found that decomposes each high-fanin gate to the limited fanin gates in the given library. Although a better decomposition may exist, due to a good ordering heuristic employed, the first successful decomposition found is typically close to the optimal in terms of area and delay.

4.3 Multi-level decompositions

If the procedure is not successful at decomposing all high-fanin gates by adding only one additional signal, the decomposition procedure is iterated to produce multiple levels of logic. After the first pass, if all gates have not been successfully decomposed, the procedure selects the decomposition for each gate which required the minimum gate size and uses its corresponding specification and implementation as input to a new iteration of the procedure. This step is repeated until an implementation is returned that either uses only gates in the library or has a minimum gate size that is no longer decreasing. In the second case, our procedure is unable to generate an implementation using the given specification and gate library. To handle this situation, either the requirements in the specification must be relaxed or carefully designed atomic gates may need to be added to the gate library.

For the *tsbm* example, if we reduce the library size to include only 2-input gates, it can no longer be decomposed with one new signal. The best decomposition that we find is the one shown in Figure 4(b) which uses only one 3-input gate which must be further decomposed. The orbital net shown in Figure 4(a) is now taken as the initial specification and the circuit shown in Figure 4(b) is the initial implementation. For this new iteration, the procedure adds an additional signal x_2 to decompose the 3-input gate. A portion of the orbital net for a decomposition that extracts the trigger signal $DSend_i$ is shown in Figure 5(a). Synthesis applied to this net results in a generalized C-implementation shown in Figure 5(b) using 3 2-input gates. Note that x_1 is a context signal in the implementation shown in Figure 4(b), and for that reason, it can move to the gate implementing x_2.

gorithm such as the one described for deterministic specifications in [2] or for more general specifications in [14] is used to determine the order of context signal transitions. Any decomposition composed of two context signals that have ordered transitions is eliminated. By taking advantage of ordering information, the number of possible decompositions for the 8-input AND gate from the *tsbm* example is reduced to only 23.

4.2 Decomposition through resynthesis

For each signal which needs to be decomposed, our procedure selects a decomposition from the set of potential decompositions that remains after applying the criterion described in the previous subsection. The original specification is then modified to incorporate a new signal for each signal being decomposed. For simplicity, we explain here the case in which the orbital net does not contain conditional behavior, or choice. We describe an example with choice later.

The procedure first adds a rising transition for each new signal to the orbital net with behavior places in its preset from the appropriate transitions on the signals in the extracted set. The timing requirements on these places have a lower bound of 0 with an upper bound derived as mentioned earlier from the maximum delay for a limited-fanin standard C-implementation. If the extracted set is composed of trigger signals, the original connections (places and transitions) between the corresponding transitions on these trigger signals and the rising (falling) transition on the signal being decomposed are replaced by a single behavior place which is added to the postset of the new rising transition. If the extracted set is composed of context signals, a constraint place with timing requirement $\langle 0, \infty \rangle c$ is added to the postset of the new rising transition and the preset of the original rising (falling) signal transition. When the falling (rising) transition of a signal also needs to be decomposed, it is done with the falling transition of the new signal using the same procedure just described. Otherwise, the falling transition of the new signal is placed between all the trigger signals for the original falling (rising) transition and the original falling (rising) transition itself.

This new specification is then resynthesized using the automatic procedure from [1] to produce a new hazard-free timed circuit implementation. If the new implementation does not have any high-fanin gates, the decomposition is successful. Otherwise, the procedure must repeat using a different decomposition for each remaining high-fanin gate.

Returning to the *tsbm* example, we apply our technology mapping procedure to the specification and implementation shown in Figure 1 with a gate library that contains 4-input AND gates, OR gates, C-elements, and generalized C-elements. One decomposition for the 8-input AND gate from the example has an extracted set that contains only the trigger signal $DSend_i$. The portion of the new orbital net corresponding to this decomposition is as shown in Figure 3(a). This new specification results in the generalized C-implementation shown in Figure 3(b). Unfortunately, this decomposition results in an implementation that requires a 7-input gate. Another possible decomposition is the one with an extracted set that contains just the context signal $\neg DSend_o$ which results in the portion of the orbital net shown in Figure 4(a). Note that the place between the new signal transition $x_1 \uparrow$ and the transition on the signal being decomposed $DReq_o \uparrow$ is now a constraint place. This decomposition produces an implementation which requires only one 2-input gate (note the generalized C-element for x_1 only requires at most two transistors in series) and one 3-input gate shown in Figure 4(b).

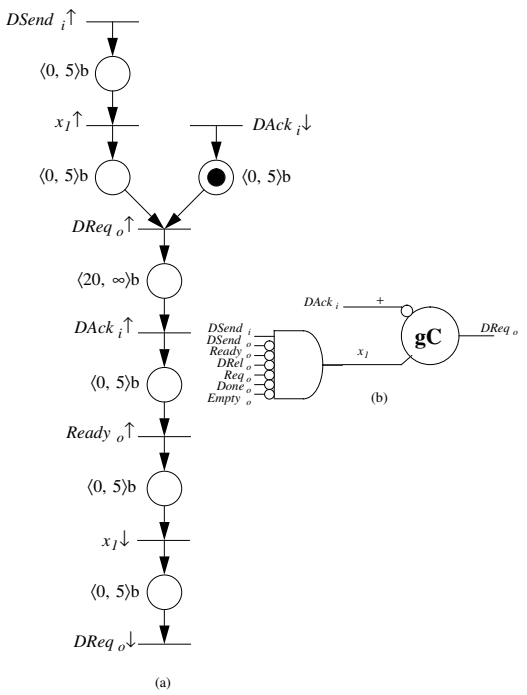

Figure 3: (a) Part of the orbital net for a decomposition using a trigger signal, and (b) corresponding generalized C-implementation.

Various cost functions can be used to evaluate different successful decompositions in terms of circuit area and delay. Because the number of different decompositions is usually small, it may be computationally feasible for the decomposition procedure to analyze each decomposition,

determines the choices of context signals which would exclude the conflicting state. Finding the smallest set of context signals to resolve all conflicts is a covering problem. Due to the implication in the entrance constraint, inclusion of certain context signals may introduce additional conflicts which must be resolved. Therefore, the covering problem is *binate*. If a conflict is detected for which there is no context signal to resolve it, or a trigger signal is not stable, it is necessary to either constrain concurrency [13], add state variables [12], or use a more general algorithm [7].

The standard C-implementation of $DReq_o$ is shown in Figure 1(b). There is one set and one reset region for $DReq_o$. The set region has trigger signals $DSend_i$ and $\neg DAck_i$ which can be read directly from the orbital net in Figure 1(a). The reset region has a single trigger signal $Ready_o$. Six context signals must be added to the cube to implement the set region, $\neg DSend_o$, $\neg Ready_o$, $\neg DRel_o$, $\neg Req_o$, $\neg Done_o$, and $\neg Empty_o$. The reset region can be implemented using just the single trigger signal. Note that when a standard C-implementation is mapped to a generalized C-element and only needs a single cube for the set and a single cube for the reset, it can be redrawn as shown in Figure 1(c). In this diagram, a "+" on a wire indicates that the signal is used in the set cube only, a "-" indicates that it is used in the reset cube only, and no annotation indicates that it is used in both.

4 Technology mapping

Given an orbital net, an unlimited fanin gate-level timed circuit implementation, and a gate library, the goal of technology mapping is to implement the circuit using only limited fanin gates found in the given library optimized to some cost function such as area or delay. The technology mapping procedure first decomposes each gate in the initial implementation with a fanin higher than that found in the gate library. Next, the partitioning step trivially identifies each signal network as a cone of logic. Finally, the matching and covering step is used to bind portions of each signal network to gates found in the library to produce an efficient implementation. It was shown in [6] that for speed-independent circuits the decomposition of high-fanin OR gates from the standard C-implementation structure can be done safely in any arbitrary manner, and that the synchronous matching/covering techniques can be used with minor modifications. These results can be easily extended to our class of timed circuits. However, For the AND gates, or cubes, care must be taken when decomposing them so as not to introduce hazards. Therefore, it is the decomposition of these cubes which the remainder of this section addresses.

Our procedure to decompose each high-fanin AND gate searches for a decomposition that uses the minimum number of logic levels. This is accomplished by adding new signals to the original specification which can be used to decompose each high-fanin gate without changing the concurrency originally specified. Each decomposition results in a modified specification which is then resynthesized to obtain a new timed circuit implementation that is guaranteed to be hazard-free. This decomposition procedure first attempts to decompose the circuit using one new signal for each high-fanin gate. If no such decomposition can be found that successfully decomposes all gates to ones found in the given library, then the specification which results in the implementation that requires the smallest fanin gates is taken as the new starting point. Using this new specification, an additional signal is added to decompose each remaining high-fanin gate. This procedure terminates either when all high-fanin gates have been successful decomposed into multi-level logic implementations, or when the minimum fanin of the best implementation and the number of gates needing to be decomposed is no longer decreasing. In the remainder of this section, we explain our decomposition procedure in more detail.

4.1 Searching the decomposition space

A decomposition of a cube is a partition of the set of trigger and context signals into two subsets: an *extracted set* and a *reduced set*. The signals in the extracted set are used as trigger signals for a transition on a new signal that is added to decompose the high-fanin gate. For a particular cube composed of n signals, there are $2^n - 1$ different decompositions. For example, the 8-input AND gate in Figure 1(b) which must be decomposed has 255 different decompositions.

Fortunately, we do not need to examine all of them as many decompositions which never lead to a successful decomposition can be safely eliminated from consideration. When two signal transitions are ordered, if the signal with the later transition is extracted as a trigger signal for the new signal transition, the signal with the earlier transition need not also be extracted. The earlier transition, if extracted, would not be a trigger signal for the new transition as two trigger signals are *never* ordered. If the signal with the earlier transition is needed in the implementation of the new signal transition, it will be as a context signal.

We use the above intuition in two ways. First, since all trigger signal transitions occur later than any context signal transition, any decomposition with an extracted set that contains both trigger and context signals from the original gate is eliminated. Second, a timing analysis al-

labeling function $s : I \cup O \rightarrow \{0, R, 1, F\}$ which returns the value of each signal and whether it is enabled, i.e.,

$$s(u) \equiv \begin{cases} 0 & \text{if } u \text{ is stable low in } s \\ R & \text{if } u \text{ is enabled to rise in } s \\ 1 & \text{if } u \text{ is stable high in } s \\ F & \text{if } u \text{ is enabled to fall in } s. \end{cases}$$

3.2 Excitation regions and quiescent states

In order to obtain our implementation, the SG is decomposed for each output signal into a collection of *excitation regions*. An excitation region for the output signal u is a maximally connected set of states in which the signal is enabled to change to a given value (i.e., $s(u) = R$ or $s(u) = F$). If the signal is rising in the region (i.e., $s(u) = R$), it is called a *set region*, and the k^{th} set region for a signal u is denoted $ER(u\uparrow, k)$. Similarly, if the signal is falling in the region (i.e., $s(u) = F$), it is called a *reset region*, and it is denoted $ER(u\downarrow, k)$. Typically, different excitation regions correspond to different output signal transitions in a high-level specification.

For each signal u, there are two sets of stable, or *quiescent states*. There is the set of states where the signal u is stable high denoted $QS(u\uparrow)$ (i.e., $QS(u\uparrow) = \{s \in \Phi \mid s(u) = 1\}$), and the set where it is stable low denoted $QS(u\downarrow)$ (i.e., $QS(u\downarrow) = \{s \in \Phi \mid s(u) = 0\}$).

3.3 Correct covers

We assume each excitation region will be implemented with a single AND gate, or *cube*, corresponding to a *cover* of the excitation region. The cover of a set region $C(u\uparrow, k)$ (or a reset region $C(u\downarrow, k)$) is a set of states for which the corresponding cube in the implementation evaluates to one. In order for a cover to lead to a hazard-free implementation, it must satisfy certain *correctness constraints* [7, 1]. These constraints guarantee that any gate in the implementation only changes when it is actively driving the output signal to change. This ensures that the transition of the gate is *acknowledged*.

First, a correct cover needs to satisfy a *covering constraint* which says that the reachable states in the cover must include the entire excitation region but must not include any states outside the union of the excitation region and associated quiescent states, i.e.,

$$ER(u*, k) \subseteq [C(u*, k) \cap \Phi] \subseteq [ER(u*, k) \cup Q(u*)]$$

where "*" indicates either "$\uparrow$" for set regions or "$\downarrow$" for reset regions.

Second, the covers of each excitation region must also satisfy an *entrance constraint* to ensure hazard-freedom. This constraint says that the cover must only be entered through excitation region states, i.e.,

$$\left[(s, s') \in \Gamma \wedge s \notin C(u*, k) \wedge s' \in C(u*, k)\right] \Rightarrow s' \in ER(u*, k)$$

To optimize the implementation, a single AND gate can be allowed to implement multiple regions. First, the procedure finds AND gate covers for each excitation region using modified correctness constraints. The covering constraint is modified to allow the cover to include states from other excitation regions, i.e.,

$$ER(u*, k) \subseteq [C(u*, k) \cap \Phi] \subseteq \left[\bigcup_l ER(u*, l) \cup Q(u*)\right]$$

The entrance constraint is similarly modified to allow the cover to be entered from any corresponding excitation region state, i.e.,

$$\left[(s, s') \in \Gamma \wedge s \notin C(u*, k) \wedge s' \in C(u*, k)\right] \Rightarrow s' \in \bigcup_l ER(u*, l)$$

An additional constraint is also now necessary to guarantee that an AND gate either covers all of an excitation region or none of it, i.e.,

$$ER(u*, l) \not\subseteq C(u*, k) \Rightarrow ER(u*, l) \cap C(u*, k) = \emptyset$$

Second, after the covers are found for each excitation region, a disjoint set of these covers must be selected to cover all regions. It is possible that no such set exists. In this case, the amount of gate sharing must be limited. A more general framework for the sharing of gates across signal networks is described in [12].

3.4 Trigger and context signals

Each cube in the implementation is composed of a set of *literals* where a literal is either an external signal or its complement. The signal corresponding to each literal is classified as either a *trigger signal* or a *context signal*. For a given excitation region, a trigger signal is a signal whose firing can cause the circuit to enter the excitation region while any non-trigger signal which is stable in the excitation region can potentially be a context signal.

Our procedure to find an optimal correct cover for a given excitation region begins with a cube consisting only of the trigger signals. If this cover contains no *conflicts*, i.e., states that violate the correctness constraints, we are done. This, however, is often not the case, and context signals must be added to the cube to remove any conflicting states. For each conflict detected, the procedure

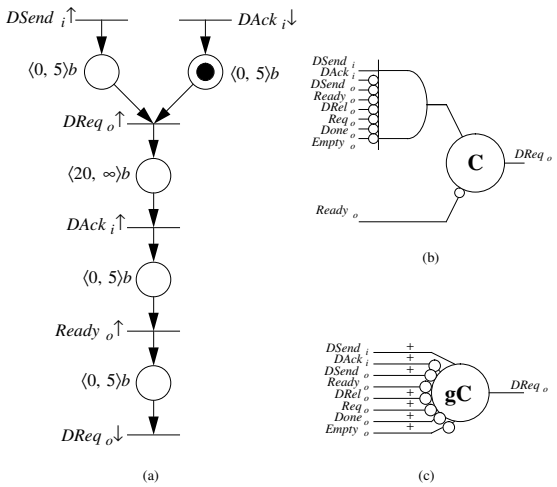

Figure 1: (a) Part of the orbital net from the *tsbm* example, (b) a standard C-implementation, and (c) a generalized C-implementation of the signal $DReq_o$.

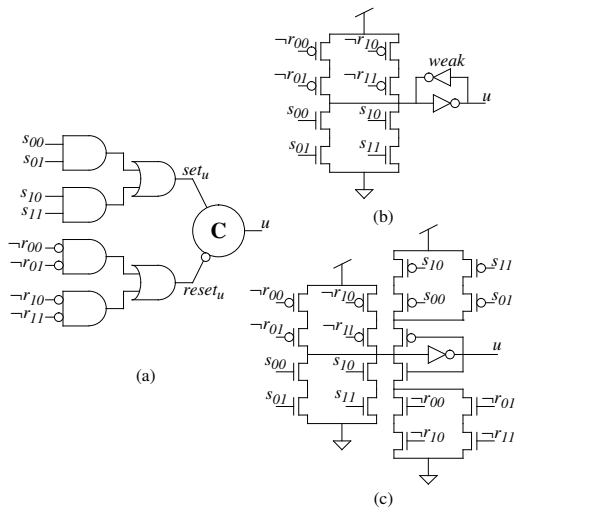

Figure 2: (a) The standard C-implementation structure, (b) a weak-feedback, and (c) a fully-static generalized C-implementation.

as shown in Figure 2(b) or as a fully-static generalized C-element as shown in Figure 2(c) [11]. When these complex gates are not available, the standard C-implementation structure is constructed using combinational gates and a C-element. If the library includes AND-OR-INVERT blocks, the sum-of-products may be mapped to them, otherwise discrete AND gates and OR gates must be used. We require that the given gate library contain at least 2-input AND gates, OR gates, and C-elements with arbitrary inverted inputs. The presence of AND-OR-INVERT blocks and generalized C-elements is optional.

It is important to note that in our timed circuit synthesis procedure, delays for transitions on output signals must be specified before the gates generating them are produced. While we believe the lower bound of the delay should be 0 as optimizations may reduce a gate to simply a wire, it is important to have a good estimate of the upper bound to produce efficient timed circuits. The solution that we propose is use an automatic analysis of the given library to derive the upper bound of the delay from the largest gate structure of the form shown in Figure 2(a) that can be built from the limited-fanin gates found in the library. Using this technique to estimate delays, however, means that when a network of gates for an output signal includes a high-fanin gate which must be decomposed to multiple levels of logic, the delay associated with transitions on this output signal may be larger than originally estimated. This increase in delay must be reflected in the specification, and it may change the resulting implementation. Since decomposition techniques for speed-independent designs do not take this into consideration, they may produce incorrect circuits when naively applied to timed circuits.

3 Synthesis

Given a specification in the form of an orbital net, the goal of the synthesis procedure is to produce a hazard-free gate-level timed circuit implementation. The synthesis procedure uses *partial order timing*, a timing analysis algorithm based on *geometric regions* and *partial orders*, to find the reachable state space for the given orbital net. By utilizing a timing analysis procedure in the derivation of the state space, the resulting *state graph* can be significantly reduced in size. For the *tsbm* example, there are 5832 states when timing is ignored, but there are only 316 states when timing is taken into account. This smaller state graph results in a significantly smaller circuit implementation. From this reduced state graph, an automatic procedure is employed to derive a hazard-free gate-level timed circuit implementation using unlimited fanin gates. This section briefly describes our synthesis procedure. For a more complete description, please see [1].

3.1 State graphs

A state graph (SG) is modeled by the tuple $\langle I, O, \Phi, \Gamma \rangle$ where I is the set of input signals, O is the set of output signals, Φ is the set of states, and $\Gamma \subseteq \Phi \times \Phi$ is the set of edges. For each untimed state s, there is a corresponding

ard is an unwanted signal transition or glitch which while filtered out by the clock signal in a synchronous design can potentially lead to a circuit malfunction in an asynchronous design. This paper addresses the problem of finding hazard-free mappings of timed circuits to limited-fanin gate libraries.

It has been shown for *fundamental mode* asynchronous circuits that synchronous technology mapping techniques can be applied with small modifications to account for hazards [4]. The fundamental-mode assumption states that inputs are allowed to change only after the circuit has settled. This assumption limits the concurrency that can be specified, and when timing analysis shows that this assumption does not hold in practice, delay elements must be added to the feedback path to guarantee that the timing constraints are satisfied, degrading the performance.

Technology mapping of speed-independent circuits has also been addressed [5, 6, 7]. The techniques employed use heuristics to investigate various decompositions, and when necessary add additional connections called *acknowledgment wire forks* to restore correctness to the decomposed implementation. These forks increase both the fanin and fanout of the gates in the implementation degrading the performance. These techniques also do not take timing into account and would produce unnecessarily conservative and possibly incorrect timed circuit implementations.

To our knowledge, the only procedure for technology mapping of asynchronous circuits that takes timing into account is the one within Berkeley's SIS [8]. This procedure derives a complex-gate implementation under the speed-independent model, and then uses synchronous technology mapping to map the design to a given gate library. The resulting implementation is then analyzed with the timing information from the library, and if hazards are detected, delay elements are added to remove them. We have shown that the implementations that are produced can be inefficient in terms of circuit area and delay due to the cost of these delay elements and the fact that timing information is neglected until late in the design process [7, 1].

In this paper, we describe an automatic procedure to map timed circuits to practical gate libraries without needing to add any delay elements. Beginning with a specification and a gate library description, our design procedure synthesizes an unlimited fanin gate-level timed circuit implementation. An automatic procedure is employed to investigate possible decompositions of any gates larger than those found in the gate library. Timing information is utilized to significantly reduce the size of the search space. From this reduced search space, each decomposition is employed to guide the resynthesis of a hazard-free timed circuit which is mapped to the given gate library. The procedure has been automated within the synthesis tool ATACS, and it has been used to map several examples.

2 Specifications and gate libraries

Our design procedure for timed circuits begins with a specification in the form of an *orbital net* and a description of the gate library. This section describes orbital nets and the types of possible gate libraries.

2.1 Orbital net specifications

An orbital net is essentially a labeled safe Petri net extended with timing [9] which can be easily derived from a high-level language [1]. An orbital net is modeled by the tuple $\langle A, P, T, F, M_0, R, L\rangle$ where A is the set of atomic actions, P is the set of places, T is the set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is the set of edges, $M_0 \subseteq P$ is the initial marking, R is an assignment of timing requirements to places, and L is a function which labels transitions with actions. A *marking* is a subset of the places. For a place $p \in P$, the *preset* of p is the set of transitions connected to p (i.e., $\{t \in T \mid (t,p) \in F\}$), and the *postset* of p is the set of transitions to which p is connected (i.e., $\{t \in T \mid (p,t) \in F\}$). For a transition $t \in T$, the presets and postsets are similarly defined (i.e., $\{p \in P \mid (p,t) \in F\}$ and $\{p \in P \mid (t,p) \in F\}$).

Timing in an orbital net is associated with a place as a timing requirement consisting of a lower bound, an upper bound, and a type (denoted $\langle l, u\rangle$*type*). There are two types of timing requirements: *behavior (b)* and *constraint (c)*. Behavior timing requirements are used to specify guaranteed timing behavior. Constraint timing requirements, on the other hand, are used to specify desired timing behavior, and they do not affect the actual timing behavior. If the timing requirement on a place is omitted, it is assumed to be $\langle 0, \infty\rangle c$. A part of the orbital net for the target-send burst-mode (*tsbm*) portion of the SCSI data transfer controller from [10] is shown in Figure 1(a).

2.2 Gate libraries

The general structure of our implementations is in the from of a standard C-implementation as depicted in Figure 2(a). In this structure, the upper sum-of-products represents the logic for the set, the lower sum-of-products represents the logic for the reset, and the result is merged with a C-element. When available in the gate library, this structure can be implemented directly in CMOS as a single compact generalized C-element with weak-feedback

Technology Mapping of Timed Circuits *

Chris J. Myers
Computer Systems Laboratory
Stanford University
Stanford, CA 94305

Peter A. Beerel
EE-Systems Department
University of Southern California
Los Angeles, CA 90089-2562

Teresa H.-Y. Meng
Computer Systems Laboratory
Stanford University
Stanford, CA 94305

Abstract

This paper presents an automated procedure for the technology mapping of timed circuits to practical gate libraries. Timed circuits are a class of asynchronous circuits that incorporate explicit timing information in the specification which is used throughout the design process to optimize the implementation. Our procedure begins with a timed specification and a delay-annotated gate library description which must include 2-input AND gates, OR gates, and C-elements, but optionally can include higher-fanin gates, AND-OR-INVERT blocks, and generalized C-elements. Our procedure first generates a technology-independent timed circuit netlist composed of possibly high-fanin AND gates, OR gates, and 2-input C-elements. The procedure then investigates simultaneous decompositions of all high-fanin gates by adding state variables to the the specification and performing resynthesis. Although multiple decompositions are explored, timing information is utilized to significantly reduce their number. Once all gates are sufficiently decomposed, the netlist can be mapped to the given gate library, taking advantage of any compact complex gates available. The decomposition and resynthesis steps have been fully automated within the synthesis tool ATACS and we present results for several examples.

1 Introduction

In recent years, there has been a resurgence of interest in the design of *asynchronous circuits* due to their ability to eliminate clock skew problems, achieve average case performance, adapt to processing and environmental variations, provide component modularity, and lower system power requirements. Traditional academic asynchronous design methodologies use unbounded delay assumptions, resulting in circuits that are verifiably correct, but sacrifice timing for simplicity, leading to unnecessarily conservative designs. In industry, however, timing is critical to reduce both chip area and circuit delay. Due to the lack of formal methods to handle timing information correctly, circuits with timing constraints usually require extensive simulation to establish confidence in the design. *Timed circuits* bridge this gap by incorporating explicit timing information into the specification and utilizing it throughout the design procedure to optimize the implementation. Timed circuits can be significantly smaller and faster than those produced using traditional methods, and they are more reliable than those produced using ad hoc methods [1]. The specification of timing constraints also facilitates a natural interaction between synchronous and asynchronous circuits.

Our previous work introduced automatic procedures for the synthesis and verification of gate-level timed circuits [2, 3, 1] and demonstrated that timed designs can be significantly smaller and faster than designs generated using other asynchronous design methodologies. The timed designs, however, are synthesized without considering explicitly the available gate library. In particular, these designs may require gates with a large number of inputs which is not practical for existing technologies. In CMOS, for example, gates with more than four transistors in series are typically considered to be too slow, and they must be decomposed. While in a synchronous design high-fanin gates can be decomposed in an arbitrary manner, in an asynchronous design decomposition must be done in such a way as to not introduce *hazards*. A haz-

*This research is supported by an NSF Fellowship and a research grant by ARPA.