# A heuristic covering technique for optimizing average-case delay in the technology mapping of asynchronous burst-mode circuits

Peter A. Beerel*
EEB 350, MC 2562
EE-Systems Dept.
USC
Los Angeles, CA 90089

Kenneth Y. Yun†
EBU1-4402, MC 0407
Dept. of ECE
UCSD
La Jolla, CA 92093

Wei-chun Chou
EEB 350, MC 2562
EE-Systems Dept.
USC
Los Angeles, CA 90089

## Abstract

*This paper presents a covering technique for optimizing the average-case delay of asynchronous burst-mode control circuits during technology mapping. The specification and NAND-decomposed unmapped network of these circuits are first preprocessed using stochastic techniques to determine the relative frequency of occurrence of each state transition and the corresponding sensitized paths through the NAND-decomposed network. We minimize the sum of the implementation's cycle times of the state transitions, weighted by their relative frequencies, thereby optimizing for average-case performance. Our results demonstrate that a 10-15% improvement in performance can be achieved with run-times comparable to synchronous techniques.*

## 1. Introduction

The principle difference between asynchronous and synchronous circuits is that asynchronous circuits are event-driven and their environment can respond as soon as an output signal changes. Thus, while synchronous circuits should be optimized for worst-case delay, asynchronous circuits should be optimized for average-case delay. While numerous tools and techniques for technology mapping of synchronous circuits [3, 8, 11, 12] optimize for worst-case delay (as well as area and/or power), the few existing techniques that target asynchronous circuits primarily focus on ensuring hazard-freedom [6, 10, 9, 5] without considering average-case performance.

*This research was funded in part by a Charles Lee Powell Foundation Research Equipment Grant for Young Faculty of the USC School of Engineering, a Zumberge Fellowship for Assistant Professors at USC, a NSF Career Award, and a gift from Intel Corporation.

†This research was supported in part by a gift from Intel Corporation.

One reason for this lack of attention is that the technology mapping step in many asynchronous design methodologies is greatly simplified by the use of specialized cell libraries. For other control-oriented synthesis methodologies, such as speed-independent, timed-circuit, and burst-mode designs, however, gate-level implementations using existing industrial standard-cell libraries are attractive. This is because designing, maintaining, and interfacing other tools with specialized cell libraries for asynchronous design is very expensive. In fact, the motivation for this work arises from the Asynchronous Instruction Decoder Project at Intel Corporation. In this project, the lack of tools and techniques forced us to manually map many control circuits onto Intel's cell library. This was both labor-intensive and inexact since we used back-of-the-envelope techniques to decide amongst various mappings.

The focus of this paper is asynchronous burst-mode control circuits which are specified with an extended burst-mode (XBM) diagram and implemented using a modified Huffman architecture [14, 13]. The extended burst-mode specification and implementation of an example, *scsi-init-send*, are given in Figure 1. This architecture has the advantage of low latency because the input-to-output path is purely combinational, *i.e.*, no explicit storage elements are used.

Note that different state transitions may occur from a given state, driving the circuit into different modes of activity. For instance, in the *scsi-init-send* example, there are two possible state transitions from state 3. The first (to state 4) represents the sending of another byte, while the second (to state 6) represents the signaling of the end of transmission. Because an output signal is often specified to change in multiple state

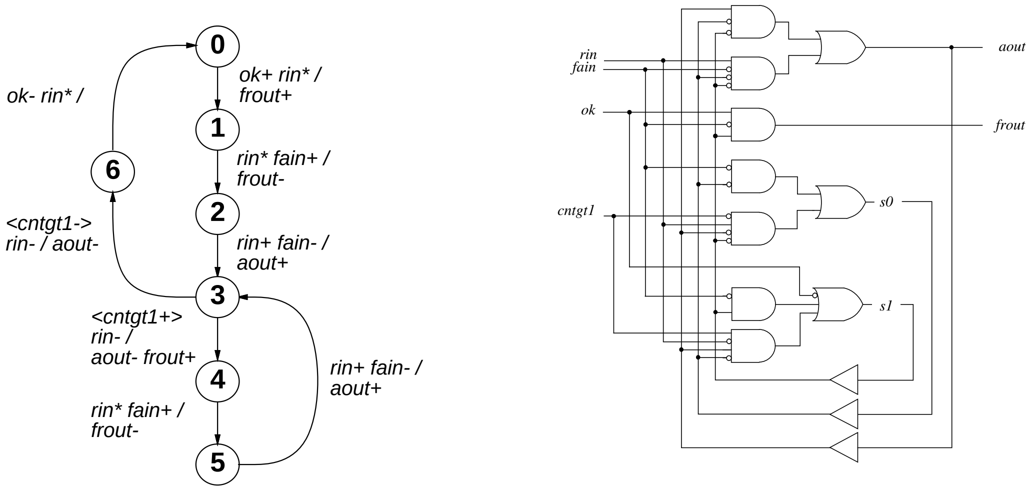


**Figure 1. Scsi-init-send example.**

transitions, its logic may contain different paths that are executed in different state transitions.

To optimize for average cycle-time, the delay along these paths should be prioritized according to the relative frequency of the associated state transition. The calculation of these priorities are based on the conditional probabilities of different state transitions. These probabilities may be given by the user or estimated through behavioral simulation of the circuit in its environment. In the *scsi-init-send* application, the conditional probabilities depend on the packet size of SCSI transfers. For example, using a reasonable packet size of 1K bytes, the conditional probabilities from state 3 would be as follows: 0.999 going to state 4 and 0.001 going to state 6. Using Markov chain analysis we determine the long run proportion of executing each state transition. The proportion of a state transition is its relative frequency and it is used to weight the delay through paths in the unmapped network that are sensitized during that state transition.

We assume that the unmapped network has been decomposed into a set of available base functions, such as 2-input NAND-gates and inverters, using a simple heuristic that optimizes the decomposition for average-case performance [1]. The paper focuses on the next step of the technology mapping algorithm which is to cover the decomposed network with available gates. Previously, we proposed an algorithm which finds the optimal solution among all possible pattern delay tradeoffs [1]. This algorithm has high computational complexity which limits its applicability to relatively small circuits (consisting of less than 500 decomposed gates) [1].

This paper presents a heuristic technique which obtains near-optimal results for small circuits with much lower run-times than previous techniques. Moreover, mapping for larger circuits (1000-2000 decomposed gates) takes under 2 minutes. The average performance improvement for a substantial benchmark of circuits is approximately 10-15% when compared to implementations mapped using a naive application of synchronous (worst-case) covering techniques.

## 2. Extended burst-mode control circuits

In each state of the machine, the circuit waits for a set of specified input transitions, referred to as an *input burst*, then changes a variety of output signals, referred to as an *output burst*. An input burst may contain *directed don't cares*, labeled with $*$, which may change monotonically but need not change and *conditionals*, enclosed in angle brackets, which represent logical levels of signals that are sampled when the remainder of the input burst arrives (for a more formal definition see [13]).

The circuit may also internally set a number of state signals. *Two-phase* state transitions have an input burst followed by an output/state variable burst where state variables change concurrently with outputs. *Three-phase* state transitions have a separate *state variable burst* that either precedes or follows the output burst [14, 13]. For the machine to operate properly, the environment must obey the *generalized fundamental-mode restriction*, meaning that a new input burst may arrive only after the circuit changes state and has settled [14]. *Cycle time* of a state transition is the delay from the last edge of an input burst to the first input edge of the next input burst required to avoid circuit malfunction (for a more formal definition see [13]).

A *pattern* is an assignment of input signals and feedback state and output signals to binary values. The *pattern arrival time* at a node in the circuit for a particular pattern is the time between the application of the pattern and the changing of the node. It is not defined for nodes that do not change in response to the particular pattern. Every burst defines a pattern whose values correspond to the values of input and feedback signals immediately after the last terminating transition of the burst arrives at the circuit. The *pattern delay* of a burst is the maximum pattern arrival time among those corresponding to output/state variables that change in response to the burst.

The cycle time of a two-phase transition equals the pattern delay of its input burst plus the delay required for the circuit to settle, which includes the feedback delay and the settling time delay. The cycle time of a three-phase transition consists of the pattern delay of the input burst, the pattern delay of the state variable burst, the feedback delay for the state variables and associated settling-time, and the feedback delay for the outputs and associated settling-time.

It has been our experience that most often no feedback delays are necessary and the settling time required is shorter than the fastest environmental response time. For the purpose of technology mapping it seems less critical to minimize these factors and thus we model them as constants *fb-delay* and *st-delay*. The cycle time, $cyc(t)$, of a two-phase state transition $t$ is then

$$\mathit{delay}(\mathit{o\text{-}burst}(t)) + \mathit{fb\text{-}delay} + \mathit{st\text{-}delay} \qquad (1)$$

and the cycle time of a three-phase state transition $t$ is

$$\begin{aligned} &\mathit{delay}(\mathit{s\text{-}burst}(t)) + \mathit{delay}(\mathit{o\text{-}burst}(t)) \\ &\quad +2(\mathit{fb\text{-}delay} + \mathit{st\text{-}delay}), \qquad (2) \end{aligned}$$

where $\mathit{delay}(\mathit{o\text{-}burst}(t))$ and $\mathit{delay}(\mathit{s\text{-}burst}(t))$ are the pattern delays of the input and state variable bursts in state transition $t$, respectively.

## 3. Pattern probabilities

Markov chain analysis is used to derive the relative frequency of occurrence of each state transition which is used to define the probabilities of each of these patterns. This analysis is similar to that used for power estimation of burst-mode circuits as well as for finding transition probabilities in finite state machines.

Under reasonable assumptions, the long term proportion of each state transition $t$, referred to as $\Pi_t$, is the unique non-negative solution to the equations:

$$\Pi_t = \sum_{t' \,:\, \mathit{sink}(t') = \mathit{source}(t)} \Pi_{t'} \cdot Pr_{XBM}(t) \qquad (3)$$

$$\sum_{t \in \delta} \Pi_t = 1, \qquad (4)$$

where $Pr_{XBM}(t)$ is the conditional probability of state transition $t$ and $\delta$ is the set of state transitions in the machine [1]. For the *scsi-init-send* example with the conditional probabilities given in the introduction, these equations show that $\Pi_t$ for each state transition $t$ has one of two weights. All state transitions in the reset cycle have weight 0.0003 and all state transitions in the main cycle have weight 0.333.

The average cycle time is then found as follows [1]:

$$\overline{\delta} = \sum_{t \in \delta} \Pi_t \cdot cyc(t). \qquad (5)$$

## 4. Optimizing average-case performance

This section presents our heuristic covering technique. The objective of the technique is to optimize the average-case performance. Specifically, the technology mapping heuristic minimizes *average pattern delay*,

$$\sum_{i \in I} w_i \cdot \mathit{delay}(i), \qquad (6)$$

subject to a given area constraint, where $I$ is the set of patterns corresponding to all input and state variable bursts, $w_i$ is the relative frequency of the associated state transition, and $\mathit{delay}(i)$ is the delay of pattern $i$.

The average pattern delay equals the average cycle time (Equation 6) of the circuit minus a constant. Because the constant does not affect the minimization, this heuristic actually minimizes the average cycle time of a burst-mode circuit subject to an area constraint.

For each node in the decomposed network, all gates that *match* the node are derived and a covering of gates that implements the network is determined. Brute-force algorithms must find every possible feasible mapping for the circuit, calculate its cost, and pick the lowest cost solution that satisfies the given area constraint. To reduce the number of feasible mappings considered, our algorithm uses dynamic programming.

The approach is similar to Chaudhary and Pedram's approach for area and delay tradeoff analysis for synchronous circuits [2]. They identify, analyze, and choose from many possible coverings representing area-delay tradeoffs. They capture these possible tradeoffs in *points* on area-delay curves that are maintained at each node in the network.

Rather than maintaining a single worst-case delay for each point, we maintain the pattern arrival times of each pattern for which that node is sensitized. Thus each point at node $n$ is of the form $\langle d_1, \ldots, d_{|I_n|}, area\rangle$, where $I_n$ is the set of patterns for which $n$ is sensitized. This set is found using Kung's algorithm [4]. An element, $d_i$, is the pattern arrival time for pattern $i \in I_n$. The element *area* is the cumulative area for the cover of this sub-cone of logic. Notice that instead of forming an area-delay curve, our set of points represents a multi-dimensional *area-delay surface*.

We describe our algorithm for circuits that consist of a forest of trees. Extensions to general DAG's are explained in [2, 1]. The algorithm contains two traversals of the circuit. First, a postorder traversal (from leaves to the roots) is used to determine the set of surface points at each node. Chaudhary and Pedram store only *non-inferior*, or *pareto*, points which represent all possibly-optimal coverings for a sub-tree rooted at this node under the constant output load assumption. Previous work extended the notion of pareto to describe surface points: a surface point is said to be *non-inferior* if there does not exist some other point $p' = \langle d'_1, \ldots, area'\rangle$ such that $d_i' \leq d_i$ for all $i$ and $area' \leq area$. Intuitively, inferior points correspond to coverings of the sub-cone rooted at $n$ that can never lead to an optimal solution; there exists a different covering which has equal or smaller delays for all input patterns and equal or smaller area. However, the number of pareto surface points that needed to be considered grew uncontrollably, limiting its applicability to small circuits. In our heuristic, only a subset of pareto surface points are stored. To define this subset we define the average pattern arrival time at a node $n$, as follows:

$$\overline{\delta}_n = \sum_{i \in I} w_i \cdot d_i. \tag{7}$$

We keep the *average-wise pareto* points on the area-delay surface of node $n$ and no more, where a point is average-wise pareto if there exists no other point on the surface which has either 1) less area and equal or smaller average pattern arrival time or 2) equal area and smaller average pattern arrival time. Our experimental results suggest that this subset of points yields near-optimal results with much lower run-times.

The algorithm to compute the area-delay surface is depicted in Figure 2. For each internal node $n$, the set of *hazard-free* matches $m$ at that node are found using an algorithm developed by Siegel and De Micheli [10]. For each match, many area-delay points are possible depending on how the cones of logic rooted at the input of the match $m$ at node $n$, referred to as $inputs(n, m)$, are mapped. Because the nodes are visited in post-order, the possibly optimal ways to implement the cones of logic rooted at $inputs(n, m)$ have already been analyzed and stored in different points on their area-delay surfaces. For each combination of input surface points, an area-delay point is computed for node $n$. The pattern arrival time $d_i$ for each pattern $i \in I_n$ is computed using the function *comp_p_time()* [1]. We use the SIS gate delay model which considers different pin-to-pin delays, distinct rise and fall times, and capacitive loading factors. The pattern delay analysis is based on the single step transition model which is more accurate than the traditional static timing models because it excludes all false paths.

Consider the example depicted in Figure 3 where a simple 2-input AND gate covers nodes D and C. There are two input patterns, one in which the AND gate rises and one in which the AND gate falls. Input B has three points and input A has 2 points. As the upper bound suggests, there are 6 possible combinations of inputs, each corresponding to a different covering of

**Algorithm 4.1 (Compute area-delay surface)**

*compute_area-delay_surface*$(n)$ {
  **foreach** *candidate match* $m$ *at node* $n$
    **foreach** *combination of inputs points pt.in_pts*
      **foreach** $i \in I$ *such that* $n \in sens(i)$
        *pt.d[i]* = *comp_p_time(pt.in_pts,i,m,n,n)*
      *pt.area* $= \sum_{k=0}^{l}$ *pt.in_pts[k].area* + *m.area*
      *insert pt into n.surf*
      *remove_inferior_pts(pt,n.surf)*
}

**Figure 2. Computing the area-delay surface.**

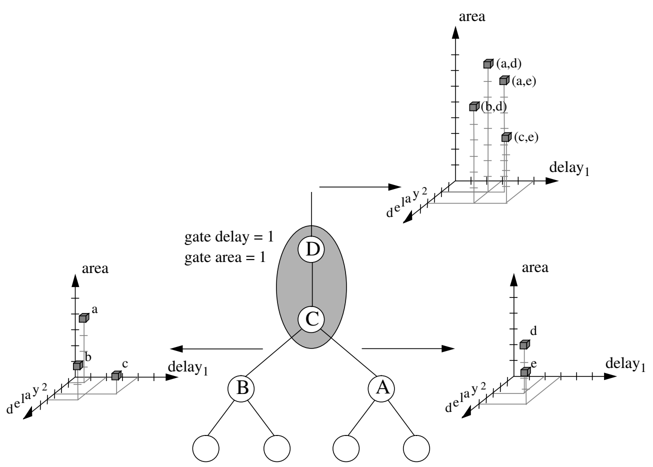


**Figure 3. Computing the surface points.**

the sub-trees rooted at A and B. Each combination determines one possible output point. For example, consider the combination of point $a$ and point $e$. The delay of the first pattern is the maximum of the delays of the first pattern in points $a$ and $e$ plus 1 for the delay of the AND gate, yielding a delay of 4. The delay of the second pattern is the minimum of the delays of the second pattern in points $a$ and $e$ plus 1 for the delay of the AND gate, yielding a delay of 2. The area for this combination is the sum of the areas of $a$ and $e$ plus 1 for the area of the AND gate, yielding 7. Assuming the pattern weights are equal, the combinations $(c, d)$ and $(b, e)$ are average-wise inferior and are not stored.

After the first traversal of the circuit, one point on the surface of each output signal is chosen. This combination of points identifies the particular technology mapping area-delay tradeoff for the entire circuit. We heuristically choose a combination of points that meets the user-specified area constraint and yields a near-optimal average pattern delay [1]. Once the combination of points is chosen, a second pre-order traversal (from roots to leaves) is performed to determine the specific gate selections that map the circuit. This second traversal is needed since the exact load of the matching gate is not known and a default load is used when initially setting the delay curve at $inputs(n, m)$. When considering a match $m$ during the pre-order phase, however, the exact load on $inputs(n, m)$ is known and used to recalculate the pattern delay. Thus, the area-delay surfaces at $inputs(n, m)$ are essentially shifted. As a result, different input points may be preferred and selected during the preorder than would otherwise be expected. This can cause a slight difference between the actual area and/or pattern arrival times than specified by the selected output point. Experimental results suggest that this difference is usually less than 10% for typical burst-mode circuits.

Our new heuristic does not change the complexity analysis given in [1] which finds the worst-case computational complexity of the algorithm to be high primarily because the number of surface points can grow exponentially with the depth of the circuit. The high computational complexity, however, is mitigated by a variety of factors. First, asynchronous control circuits tend to be relatively small (less than 1K gates) because asynchronous systems are typically designed using a distributed control paradigm. Second, the number of patterns of interest for each output tends to be small. Third, for each pattern only the gates on the sensitized paths (which tend to be a small subset of all paths) have corresponding pattern arrival times. And, finally, most derived area-delay points are average-wise inferior and therefore dropped.

## 5. Preliminary Results

We implemented a preliminary version of our covering algorithm in an extension to POSE, which is an extended version of SIS that contains the code for Chaudhary and Pedram's technology mapping algorithm [2] and other optimizations targeting low-power design.

We conducted experiments on a suite of benchmark circuits on a SPARC 20 model 502 with 128 Megabytes of memory. We applied both worst-case [2] and average-case methods on a NAND-decomposed network of the circuit that has been optimized for average-case delay in Table 1. We report results using both the previous method described in [1] and our new heuristic method described in this paper. For all experiments, we used the "lib2" gate library included in the SIS package. Due to lack of time, De Micheli and Siegel's algorithm was not used to determine which library gates are hazardous. Instead, all gates were assumed to be hazard-free.

The results demonstrate that the application of our covering algorithm can yield up to 10% to 15% improvement over the traditional worst-case covering. As expected, the circuits with widely varying state transition frequencies, *i.e., scsi-init-send, dramc, trcv, ircv, tsend, stetson-p1, isend, scsi* and *cache-ctrl*, show significantly greater improvements than the circuits with uniform state transition frequencies, *i.e., merge, bufc-trl, stetson-p3, q42* and *binary-cntr*.

Compared to previously reported results, the run-time reductions are dramatic. Our new technology mapping heuristic can complete on many lager circuits for which our previous average-case optimization algorithm times out. Circuits of up to 2000 decomposed gates (the largest circuits available) have been successfully mapped in under 2 minutes. The results also show that these run-times are comparable to those of traditional worst-case covering.

Moreover, in all circuits for which both algorithms complete, the new heuristic achieves less than 5% worse results than the previous algorithm. In fact, for some circuits it did slightly better, presumably due to random differences caused by the unknown load approximation.

## 6. Conclusions

This paper describes an efficient technology-mapping covering heuristic for optimizing average-case performance of burst-mode circuits. The results demonstrate that average-case covering typically has comparable run-times to synchronous techniques and yields 10-15% improvement in performance.

| Preliminary Results of Covering Technique | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Circuit Description | | | | | | Ave. Pattern Delay (ns) | | | CPU Times |
| | # | # | # | # | # | Worst-case | Ave-case | | Worst-case/Heuristic/ |
| Name | PIs | POs | States | Patts. | Gates | | Heuristic | Previous | Previous (secs) |
| merge | 8 | 3 | 7 | 8 | 31 | 2.0 | 1.63 | 1.68 | 0.2/0.1/0.1 |
| bufctrl | 5 | 3 | 3 | 3 | 36 | 2.84 | 2.80 | 2.80 | 0.3/0.1/0.1 |
| stetson-p3 | 7 | 3 | 8 | 11 | 40 | 1.69 | 1.66 | 1.66 | 0.3/0.2/0.1 |
| q42 | 5 | 3 | 4 | 4 | 45 | 3.54 | 3.49 | 3.49 | 0.3/0.2/0.2 |
| binary-cntr | 11 | 7 | 32 | 32 | 72 | 5.49 | 5.11 | 5.11 | 1.6/0.8/0.98 |
| dramc | 13 | 7 | 12 | 14 | 135 | 2.96 | 2.80 | 2.75 | 1.2/0.5/3.0 |
| scsi-init-send | 10 | 4 | 7 | 10 | 142 | 4.43 | 3.97 | 3.88 | 0.6/0.3/0.3 |
| pe-send-ifc | 12 | 5 | 11 | 14 | 172 | 4.31 | 4.22 | 4.17 | 1.4/0.7/7.6 |
| trcv | 14 | 7 | 16 | 30 | 401 | 7.88 | 6.78 | 6.93 | 4.1/3.6/203.6 |
| ircv | 14 | 7 | 16 | 30 | 451 | 8.58 | 7.40 | 7.33 | 5.1/3.0/132.9 |
| tsend | 18 | 9 | 22 | 42 | 856 | 12.13 | 10.57 | time-out | 10.23/12.1/time-out |
| stetson-p1 | 28 | 17 | 26 | 31 | 932 | 7.36 | 6.34 | time-out | 10.7/10.8/time-out |
| isend | 28 | 17 | 26 | 31 | 1423 | 13.22 | 11.76 | time-out | 23.8/27.2/time-out |
| scsi | 19 | 11 | 71 | 93 | 1741 | 10.9 | 9.70 | time-out | 24.3/72.8/time-out |
| cache-ctrl | 37 | 20 | 38 | 49 | 2092 | 13.0 | 12.09 | time-out | 29.0/61.9/time-out |

**Table 1. Preliminary results of our covering algorithms for minimizing average cycle-time.**

We also believe that these technology mapping algorithms are applicable to other fundamental mode design styles, such as Nowick's UCLOCK method [7].

## References


[1] P. A. Beerel, K. Yun, and W.-C. Chou. Optimizing average-case delay in technology mapping of burst-mode circuits. In *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 1996.

[2] K. Chaudhary and M. Pedram. Computing the area versus delay trade-off curves in technology mapping. *IEEE Transactions on Computer-Aided Design*, pages 1480–1489, Dec. 1995.

[3] K. Keutzer. DAGON: Technology binding and local optimization by DAG matching. In *24th Design Automation Conference*, pages 341–347. IEEE/ACM, 1987.

[4] D. Kung. Path sensitization in hazard-free circuits, 1995. In collection of papers of the *ACM International Workshop on Timing Issues in the Specification of and Synthesis of Digital Systems.*

[5] 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, Jan. 1995.

[6] C. J. Myers, P. A. Beerel, and T. H.-Y. Meng. Technology-mapping of timed-circuits. In *2nd Working Conference on Asynchronous Design Methodologies*, May 1995.

[7] S. Nowick and B. Coates. UCLOCK: automated design of high-performance unclocked state machines. In *Proceedings of the 1994 IEEE International Conferen ce on Computer Design: VLSI in Computers and Processors*, October 1994.

[8] R. Rudell. *Logic Synthesis for VLSI Design.* PhD thesis, U. C. Berkeley, Apr. 1989. Memorandum UCB/ERL M89/49.

[9] P. Siegel and G. DeMicheli. Decomposition methods for library binding of speed-independent asynchronous designs. In *IEEE/ACM 1994 ICCAD Digest of Technical Paper*, 1994.

[10] P. Siegel, G. D. Micheli, and D. Dill. Automatic technology mapping for generalized fundamental-mode asynchronous designs. In *Proc. ACM/IEEE Design Automation Conference*, pages 61–67, June 1993.

[11] H. J. Touati, C. W. Moon, R. K. Brayton, and A. Wang. Performance-oriented technology mapping. In W. J. Dalley, editor, *6th MIT Conference on Advanced VLSI Conference*, pages 79–97, 1995.

[12] C.-Y. Tsui, M. Pedram, and A. M. Despain. Power efficient technology decomposition and mapping under an extended power consumption model. *IEEE Transactions on Computer-Aided Design*, 13(9):1110–1122, 1995.

[13] K. Y. Yun. *Synthesis of Asynchronous Controllers for Heterogeneous Systems.* PhD thesis, Stanford University, Aug. 1994.

[14] K. Y. Yun, D. L. Dill, and S. M. Nowick. Synthesis of 3D asynchronous state machines. In *International Conference on Computer Design, ICCD-1992.* IEEE Computer Society Press, 1992.