Algorithms to Simplify Multi-Clock/Edge Timing Constraints

Veerapaneni Nagbhushan, C.Y. Roger Chen
Dept. of EE & CS, Syracuse University, Syracuse, NY
vnagbhushan@yahoo.com, crchen@syr.edu

Abstract

The use of multiple clocks has become a common practice in modern microprocessor design. With multiple clocks, the timing specifications have become complicated and tend to go beyond the ability of single-clock based CAD tools. This paper first introduces the concept of timing specification transformation. Then, this paper describes algorithms for transforming an interface timing specification with multiple clocks/edges into an equivalent specification with a single clock/edge for combinational circuit blocks. It formulates a new optimization problem, which is important but has never been addressed by CAD researchers. It identifies conditions under which this transformation can be performed efficiently without any loss of timing budget. The algorithm can be used to simplify the constraints to drive many synthesis and optimization algorithms.

1. Introduction

Accurate specification of the timing requirements of a block of logic is essential to drive downstream algorithms such as timing driven synthesis, placement, sizing, power optimization, routing, etc. The timing specifications are usually provided in the form of arrival/valid times for signals at the input pins of the block and required arrival times for signals at the output pins. The arrival and required times are usually specified with reference to some global ideal (not physical) event such as the rising/falling edge of an ideal clock.

Due to increasing demands on frequency and power, modern digital chips utilize several clocks within their circuitry. In many cases, the block whose timing has to be specified ends up having interface signals (pins) whose timing is specified with respect to different clocks and/or different edges. In fact, each pin can have several such events.

Most logic synthesis and technology mapping algorithms, that address timing constraints directly (as opposed to just minimizing the timing, which could lead to significant over-design) do so by directly or indirectly propagating arrival times [1,2,5,8,9] or delays [3,4,7]. Some ([10]) assume a single clock. All these algorithms implicitly deal with only one constraint clock/edge or assume that there is a dominant clock which is used to drive the solution (i.e. a pure delay model). These algorithms can not be easily/efficiently modified to handle multiple clocks/edges.

Hence there is a need to develop an algorithm to (1) eliminate the reference to multiple reference clocks/edges and (2) reduce multiple events on a pin into a single event without relaxing the intent of the original timing specs. Then, the simplified specs can be used to drive downstream tools such as synthesis while guaranteeing that the resulting circuit will satisfy the original timing specs in its original environment. Even when the synthesis/optimization algorithm can deal with multiple clocks/edges, this algorithm can be used to reduce/simplify the timing constraints up front so that the algorithm does not have to deal with this complexity in its inner loops. This can improve the run-time of the algorithm. [6] addresses the problem of generation of timing constraints to guide the re-design of portions of combinational logic. However, the concept of simplifying the timing specifications from multiple clock/edge to single clock/edge has not been systematically addressed so far.

The outline of the rest of the paper is as follows. Section 2 defines the problem and introduces the concept of specification transformation. Section 3 translates the specification into a timing budget graph and proves that if certain conditions on the graph are satisfied, an efficient and optimal solution exists. Section 4 describes solutions to the general and restricted problem. Section 5 shows results based on current industrial processor designs.

1-4244-1258-7/07/$25.00 ©2007 IEEE

2. Problem Definition

The connectivity interface of a combinational block of logic, B, is specified by a set of interface pins $P = \{p_1, p_2, \cdots, p_n, p_{n+1}, p_{n+2}, \cdots, p_{n+m}\}$. For the rest of the paper, let $\{p_1, p_2, \cdots, p_n\}$ be the input pins and let $\{p_{n+1}, p_{n+2}, \cdots, p_{n+m}\}$ be the output pins. Functionally, each output pin is a Boolean function of one or more input pins:

$p_{n+i} = f_{n+i}(p_1, p_2, \cdots, p_n), 1 \le i \le m$

Let F be the vector $(f_{n+1}, f_{n+2}, \ldots, f_{n+m})$.

2.1. Timing specification description

Each timing specification (or event) specifies a delay (in time units) before/after the rise/fall edge of an ideal clock. Formally, a timing specification is a 4-tuple (*delay*, before|after, *clock*, rise|fall) where *clock* is an element in *global_clock_set*, the set of all clocks in the design.

It is assumed that the clock cycle times are harmonically related (i.e. there is an overall cycle time which is an integral multiple of each cycle time). There is no restriction on their duty cycles.

Associated with each input pin are one or more arrival times which specify the latest times after which the input signal is valid (and can be sampled/used by the block). Hence "100 *A clk_a R*" means that the signal is needed 100ps after rise of *clk_a*. Similarly, for each output pin, one or more required arrival times are specified, which denote the earliest time when the signal is required at that pin (to be sampled by the receiving logic). Hence "100 *B clk_a F*" specifies that the signal is needed 100ps before the falling edge of *clk_a*.

Each input pin p_i is associated with a set of arrival times $A_i = \{a_{i,1}, a_{i,2}, \ldots\}$ and each output pin p_j is associated with a set of required arrival times $R_j = \{r_{j,1}, r_{j,2} \ldots\}$. Let A be the vector $(A_1, A_2, \ldots, A_n)$ and R be the vector $(R_{n+1}, R_{n+2}, \ldots, R_{n+m})$. Thus a combinational block B can be fully specified by its functional description F and its arrival and required times A and R. Hence, $B = (F, A, R)$

2.2. Transformation of timing constraints

Let us define *event timing budget*, $d_{a,r}$, between an arrival specification a at an input pin and a required specification r at an output pin as the interval which is available for logic computation between those two specifications. Since each pin may have multiple timing specifications, let us define the *pin timing budget*, $c_{i,j}$ between input pin i and output pin j, as the smallest event timing budget among the arrival-required spec pairs:

$$c_{i,j} = \underset{(a,r) \in A_i \times R_j}{MIN} \{d_{a,r}\}$$

The arrival times vector A and the required times vector R effectively specify the time budgets for all paths in the circuit block and set the constraints for timing-driven tools.

A timing specification (A, R) can be transformed into another specification (A', R'). Let us define such a transformation to be *valid* if (A', R') satisfies all the constraints imposed by (A, R) and denote it as $(A', R') \supseteq (A, R)$. Note that a transformation can be valid even though the numeric values for the arrival/required specs may have changed.

Theorem 1: $(A', R') \supseteq (A, R)$ if

$$\forall (i = 1 \cdots n,\ j = n+1 \cdots n+m)\ \ c'_{i,j} \le c_{i,j}$$

where $c_{i,j}$ and $c'_{i,j}$ are the pin timing budgets in (A, R) and (A', R') respectively.

The proof is left out due to space limitations.

We can have three cases:

- Case 1: As long as the pin timing budget for each input-output combination in the circuit is maintained during the transformation, there is no loss from a timing perspective. Let us define such transformations as *non-reducing transformations*. If (A', R') is simpler than (A, R) by having fewer events or fewer clocks, this is a useful transformation.
- Case 2: Some transformations may reduce the pin timing budget for one or more input-output combinations. Such a transformation is still valid (from Theorem 1). Let us call such transformations as *reducing transformations.* In those cases, the paths whose budgets have been reduced will have to be synthesized/placed with less timing budget than the original. Hence it is preferable to reduce the budgets as little as possible.
- Case 3: Any transformation which increasing the pin timing budget on any path violates Theorem 1 and is hence not a valid transformation.

2.3. Problem statement:

Given a combinational block $B = (F, A, R)$, find an equivalent specification (A', R') such that:

1. All the arrival and required events in A' and R' refer to one and only one clock,
2. $|A'_i| = 1 = |R'_i| \quad \forall 1 \le i \le n < j \le n+m$
   i.e. each pin has only 1 arrival/required spec,
3. $(A', R') \supseteq (A, R)$
4. budgets are reduced minimally, if at all.

3. Formulation

For simplicity, let us consider a subset of the general problem where each pin has only one arrival or required specification. Also, without loss of generality, let us set the delay in each arrival and required specifications to 0 (non-zero delays simply result in scalar additions/subtractions to the valid/required times).

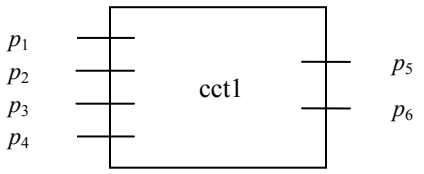


Figure 1. Block inputs and outputs.

Consider the example in Figure 1 with 4 inputs and 2 outputs.

$p_5 = f_5(p_1, p_2, p_3)$, $p_6 = f_6(p_3, p_4)$

$F = (f_5, f_6)$

Each input pin has one arrival and each out pin has one required spec.:

$A = [\{a_1\}, \{a_2\}, \{a_3\}, \{a_4\}]$, $R = [\{r_5\}, \{r_6\}]$

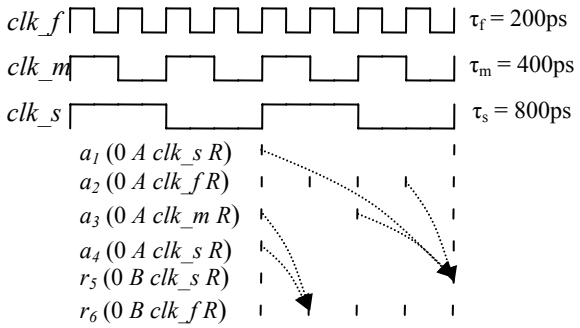


Figure 2. Timing spec for cct1

Consider a clock system with 3 clocks, fast, medium and slow, as shown in Figure 2 for cct1.
Therefore, *global_clock_set* = {*clk_f*, *clk_m*, *clk_s*}
Let τ_c be the cycle time of clock *clk_c*. In the example, τ_f = 200ps, τ_m = 400ps, τ_s = 800ps.

Figure 2 shows the arrival and required specifications. The pin timing budget for each input-output pair is (as shown by the dotted lines):
$c_{1,5} = \tau_s$ = 800ps, $c_{2,5} = \tau_f$ = 200ps, $c_{3,5} = \tau_m$ = 400ps
$c_{3,6} = \tau_f$ = 200ps, $c_{4,6} = \tau_f$ = 200ps

3.1. Timing Budget Graph

Based on the pin timing budgets ($c_{i,k}$) each output p_k induces an ordering (with distances) among its inputs, as shown in Figure 3. Let us call this a *timeline,* T_k , for the output pin p_k.

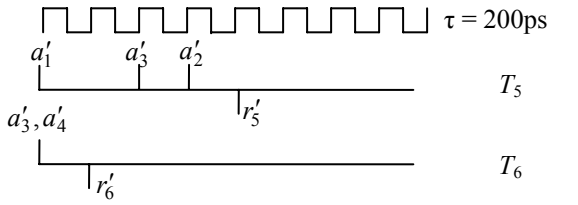


Figure 3. Timelines for cct1

They can be combined into a *timing budget graph* (*TBG*) as shown in Figure 4. The TBG is a directed weighted multigraph $G = (V, E)$ where

$V = \{a'_i \mid 1 \le i \le n\} \cup \{r'_j \mid n+1 \le j \le n+m\}$ and

$E = \{(u, v, dist_{u,v}) \mid u, v \in V\}$.

Recall that n and m are the number of input & output pins respectively. a'_i and r'_i represent the solution (single arrival and required specification on inputs p_i and outputs p_j respectively). $dist_{u,v}$ is the distance between u and v on the timeline.

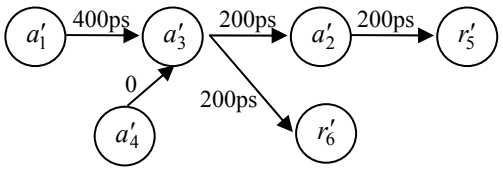


Figure 4. Timing budget graph for cct1

Note that there can be more than one edge between a pin pair. *G* can have cycles. We can have a forest of unconnected graphs.

Theorem 2: If the following two conditions on the timing budget graph, *G*, are met, there exists a non-reducing transformation of the timing specification into an equivalent single clock specification:

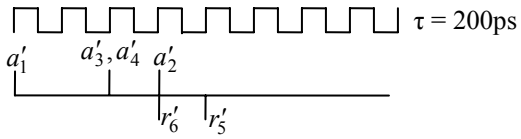
1. *G* does not have cycles and,
2. *G* does not have multiple paths with different path weights between any two vertices

The proof is left out due to space limitations.

Note that any timing budget graph that satisfies the conditions in Theorem 2 is effectively a weighted directed acyclic graph (not a multi-graph).

3.2. Examples

Since the graph in Figure 4 satisfies both the conditions of Theorem 2, the problem can be reduced to a single clock specification as shown in Figure 5. The single clock, *clk_c*, has a cycle time $\tau_c = \tau_s = 800$ps. The arrival and required specifications are now delayed appropriately to maintain their relative path timing budgets. The solution is optimal and no budgets need to be reduced. The linear time offsets and the corresponding timing specification (in parenthesis) are shown.



$a'_1 = 0$ (0 *A clk_s R*)

$a'_2 = 600$ps (600 *A clk_s R*)

$a'_3 = a'_4 = 400$ps (400 *A clk_s R*)

$r'_5 = 800$ps (0 *B clk_s R*)

$r'_6 = 600$ps (200 *B clk_s R*)

Figure 5. Solution for cct1

Figure 6 shows a situation where the TBG has multiple paths with different path weights (0 and 200ps) between a'_1 and a'_2.

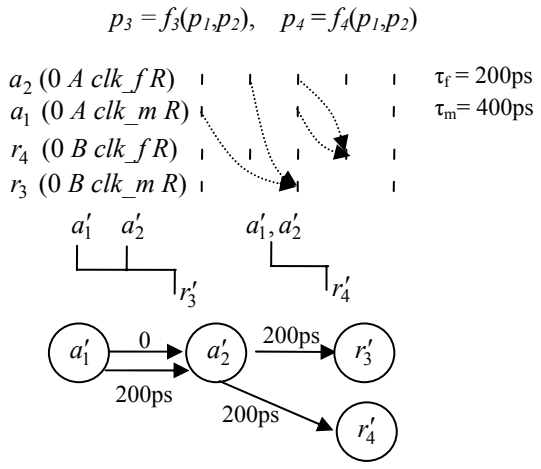


Figure 6. Circuit with conflicts (cct2)

Figure 7 shows a TBG with cycles. It can be shown that this can happen only if some of the arrival or required specifications are on different clock edges (i.e. rise/fall).

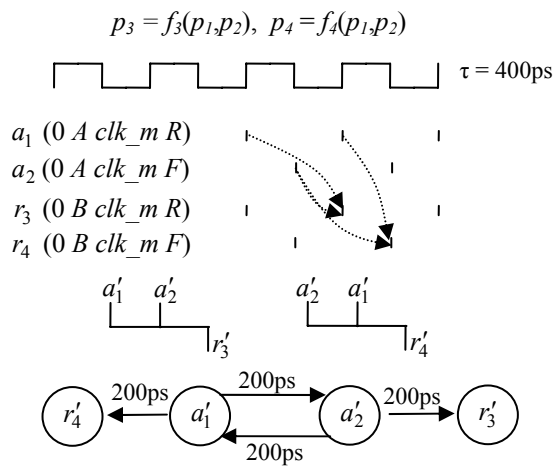


Figure 7. Circuit with cycles in TBG (cct3)

Neither cct2 nor cct3 have optimal solutions as they violate Theorem 2.

4. Solution

We first describe a quick solution to solve the problem when the conditions of Theorem 2 are satisfied. We then describe a general solution framework based on numerical programming to solve the problem when these conditions are not satisfied.

4.2 Quick solution

If the timing budged graph (TBG) satisfies the conditions specified in Theorem 2, the problem can be solved quickly and optimally using graph traversal. Recall that such a TBG is a weighted directed acyclic graph $G = (V, E)$ where $E = \{(u, v, dist) \mid u, v \in V\}$.

Define Modify(G) = $G' = (V, E')$, where

$$E' = E \cup \{(v, u, 0 - dist) \mid (u, v, dist) \in E\}$$

Thus G' is obtained by adding reverse edges with negative weight to G.

Procedure Quick_Solution

Input: TBG $G = (V, E)$ which satisfies Theorem 2.

Output: offset[v], forall $v \in V$

1. Topologically sort $G = (V,E)$ into a linear ordering of vertices. Let the first vertex be v_1.
2. **foreach** $v \in V$ **do** $seen[v] \leftarrow FALSE$
3. $offset[v_1] \leftarrow 0$
4. $seen[v_1] \leftarrow TRUE$

5. $enqueue(Q, v_1)$
6. $G'(V, E') \leftarrow Modify(G)$
7. **while** $(Q \neq \phi)$ **do**
8. $u \leftarrow dequeue(Q)$
9. **foreach** $(u, v, dist) \in E'$ **do**
10. **if** $seen[v]$ = FALSE **then**
11. $enque(Q,v)$
12. $seen[v] \leftarrow$ TRUE
13. $offset[v] \leftarrow offset[u] + dist$

The algorithm converts the TBG into a bi-directional graph, G'. Lines 7-13 perform a breadth first traversal of G' using a FIFO/queue Q; the offset is propagated whenever an unseen vertex is encountered. The initial topological sort is not necessary; it is done to make most of the offsets positive, which is aesthetically preferable.

If the TBG is actually a forest of unconnected graphs, the above algorithm can be performed on each graph in the forest.

The resulting offsets are the timing assignments in the single clock formulation.

Let us illustrate this algorithm on the TBG in Figure 4. Let a'_4 be the starting vertex v_1 (line 1). The modified graph G' is similar to the TBG, except that the edges are bi-directional – the added/reverse edges have negated edge weights. In the first pass of the BFS loop (lines 7-13), a'_4 is dequeued and its unseen neighbor, a'_3, is enqueued and its offset set to 0. In the second pass, a'_3 is dequeued and its unseen neighbors, a'_2, a'_1 and r'_6, are enqueued and their offsets set to +200ps, -400ps and +200ps respectively. Note that since we traversed a reversed edge to get to a'_1, we use -400ps as the offset increment. In the third pass, a'_2 is dequeued and its unseen neighbor, r'_5, is enqueued and its offset is set to 400ps. The remaining iterations of the BFS loop unload the queue and don't discover any new vertices.

The topological sort in line 1 can be done in O(V+E) time. Creating the modified graph in line 6 can be done in O(V+E) time (using its acyclicity). The resulting graph G' now has 2|E| edges. The BFS loop and offset propagation in lines 7-13 can be done in O(V+E) time. Hence the entire algorithm runs in O(V+E) time.

4.1. General Solution

4.1.1. Constraints. Irrespective of whether the TBG has loops or not, the problem can be modeled as a numerical programming problem with linear constraints and quadratic objective function as follows:

Constraints: $r'_j - a'_i \leq c_{i,j} \ \forall (i,j) \mid p_j = f(p_i)$

$c_{i,j}$ are the known pin timing budgets between input pin p_i and output pin p_j where there is a functional path from p_i to p_j. The inequality implies that the timing budget may be reduced, if needed, but can't be increased (from Theorem 1).

4.1.2 Optimization functions. Several solutions are possible for the linear constraint problem specified above. However, some of them will reduce the pin timing budgets more than others.

For each input-output pin pair p_i, p_j where $p_j = f(p_i)$, let us define *budget reduction* as

$b_{i,j} = c_{i,j} - (r'_j - a'_i)$

We would ideally like to obtain a non-reducing specification, where $\sum b_{i,j} = 0$

Hence we have

Objective1: Minimize $\sum b_{i,j}$ for all input-output pin pairs p_i, p_j.

This works well in cases where non-reducing transformations are possible (i.e. when Theorem 2 is satisfied). However, when some budgets have to be reduced to meet the constraints, Objective1 may behave erratically. Since all budget reductions are linearly summed in Objective1, it can result in reducing some path budgets by a large amount (sometimes zeroing them out) even when it is not necessary. Specifying a zero time budget for a pin pair means that the circuit along that path can not be realized. Hence it should be avoided. Using a quadratic objective function, as in Objective2 below, results in a more even reduction of budgets between pin-pairs.

Objective2: Minimize $\sum \left(b_{i,j}\right)^2$ for all pin pairs p_i, p_j.

For cct2 in Figure 6, we can have the following formulation and solution:

Constraints:

$$r'_4 - a'_1 \leq 200, \quad r'_4 - a'_2 \leq 200$$
$$r'_3 - a'_1 \leq 400, \quad r'_3 - a'_2 \leq 200$$

Objective: Minimize: $\left(r'_3 - a'_2 - 200\right)^2 +$

$$\left(r'_4 - a'_1 - 200\right)^2 + \left(r'_4 - a'_2 - 200\right)^2 + \left(r'_3 - a'_1 - 400\right)^2$$

Results:

$a'_1 = 0,\ a'_2 = 0,\ r'_4 = 200,\ r'_3 = 200$

Note that $a'_1 \rightarrow r'_3$ budget is reduced.

Similarly, for the circuit in Figure 7 with the cyclic TBG, we have the following formulation and results:
Constraints:

$$r'_3 - a'_1 \le 400\,, \quad r'_3 - a'_2 \le 200$$
$$r'_4 - a'_1 \le 200\,, \quad r'_4 - a'_2 \le 400$$

Objective: Minimize: $(r'_4 - a'_2 - 400)^2 +$

$$(r'_3 - a'_1 - 400)^2 + (r'_3 - a'_2 - 200)^2 + (r'_4 - a'_1 - 200)^2$$

Results:

$$a'_1 = 0,\ a'_2 = 0,\ r'_4 = 200,\ r'_3 = 200$$

Note that $a'_1 \to r'_3$ and $a'_2 \to r'_4$ budgets are reduced.

It should be noted that other optimization functions or methods are also possible.

5. Results

The proposed algorithms have been implemented and run in the context of a layout based logic re-synthesis flow for high performance processor designs on 65nm process technology. Purely combinational regions of physically close, timing critical cells were extracted using a clustering algorithm.

Table 1 shows some results for several regions using the numerical programming formulation with quadratic objective function. The columns are the size (number of standard cells), number of input and output pins and the number of unique reference clocks/edges (before applying the algorithm) for the regions. The last column is the budget reduction percentage ($\sum b_{i,j}$ as a fraction of $\sum c_{i,j}$)for all the input/output pin pairs. In all cases, the number of reference edges was reduced to one.

Table 1. Results

	size	ipins	opins	Refs before	Budget reduction
Cct1	328	257	132	3	0%
Cct2	437	367	265	3	0%
Cct3	1577	883	542	5	0%
Cct4	523	345	367	3	0.2%
Cct5	984	475	325	3	0.02%

We can see from the results that it is practical to apply this algorithm without reducing the overall budgets in a big way.

6. Conclusions and Future Work

We show a method to formulate the multi-clock/edge specification problem and reduce the specification to a single clock/edge. We convert the timing specification problem into a timing budget graph and prove that when a set of conditions are satisfied, the timing specification can be reduced efficiently and without any loss of time budget. When these conditions are not met, we formulate an optimization problem and solve it to yield the single clock/edge specification. Finally, we show that these algorithms produce reasonable results on current processor circuits.

Future work could address rise and fall timing specs for the signals (not the reference edge, which we have already considered), better optimization functions, allow sequential elements inside the block and handle timing overrides.

Acknowledgement

This work was done when the first author was at Intel Corporation, Santa Clara, California.

References

[1] W.Gosti, S.R.Khatri and A.L.Sangiovanni-Vincentelli, "Addressing the timing closure problem by integrating logic optimization and placement", *International Conference on Computer Aided Design*, November 2001.

[2] H.J.Touati, C.W.Moon, R.K.Brayton and A. Wang, "Performance-oriented technology mapping", *Proc. 6th MIT Conf. Advanced Research in VLSI*, 1990.

[3] K.Chaudhary and M.Pedram, "A Near Optimal Algorithm for Technology Mapping Minimizing Area under Delay Constraints", *Design Automation Conference*, 1992

[4] H.J.Touati, "Performance-Oriented Technology Mapping". PhD thesis, Univ. of CA, Berkeley, 1990, Memorandum No. UCB/ERL M90/109.

[5] C-S.Wang and C.Yeh, "Performance-driven technology mapping with MSG partition and selective gate duplication", *ACM Transactions on Design Automation of Electronic Systems*, v.11 n.4, October 2006

[6] N.Weiner, and A.Sangiovanni-Vincentelli, "Timing Analysis in a Logic Synthesis Environment", *26th Conference on Design Automation*, 1989.

[7] R.Shelar, P.Saxena, X.Wang, S.Sapatnekar, "An efficient technology mapping algorithm targeting routing congestion under delay constraints", *International Symposium on Physical Design*, April 2005.

[8] A.Srivastava, R.Kastner, M.Sarrafzadeh, "Timing driven gate duplication: complexity issues and algorithms", *International Conference on Computer-Aided Design*, November 2000.

[9] A.Oliveira, R.Murgai, "An exact gate assignment algorithm for tree circuits under rise and fall delays", *International Conference on Computer-Aided Design*, November 2000.

[10] B.F.Mo, R.Brayton, "A Timing-Driven Module-Based Chip Design Flow", 41st Conference on Design Automation (DAC'04), June 2004.