Timing Model Reduction for Hierarchical Timing Analysis

Shuo Zhou*, Yi Zhu†, Yuanfang Hu†, Ronald Graham†,
Mike Hutton ‡, Chung-Kuan Cheng‡
Synopsys Corp. *, Department of Computer Science and Engineering University of California, San Diego†,
Altera Corp.‡

***Abstract*— In this paper, we propose a timing model reduction algorithm for hierarchical timing analysis based on a biclique-star replacement technique. In hierarchical timing analysis, each functional block is characterized into an abstract timing model. The complexity of analysis is linear to the number of edges in the abstract timing model for timing propagation. We propose a biclique-star replacement technique to minimize the number of edges in the timing model. The experiments on industry test cases show that by allowing acceptable errors, the proposed algorithm can largely reduce the number of edges in the timing model.**

***Index Terms*— Hierarchical Timing Analysis, Biclique-star Replacement**

I. Introduction

In hierarchical timing analysis, a design is divided into multiple blocks and each block is characterized into an abstract timing model. For linear delay model, we add delays of edges on a path linearly to get path delay and ignore the second order effects such as nonlinear functions of slew rates and output loads. We can use linear delay model in FPGA timing analysis or at high level optimizations. As a result, the timing calculation can be separated according to the boundary of the partitions. Assume the timing relation inside each block is fixed. During static timing analysis, we do not need to go through the details in the blocks but use the pre-calculated timing models for timing calculation. The analysis complexity is linear to the number of edges in the abstract timing model for timing propagation. Therefore, we should minimize the number of edges in the timing model to improve the analysis efficiency.

There are some previous works related to the timing model minimization. Some techniques start from the timing graph of the block, and iteratively reduce the number of edges in the graph using graph transformations [1], [2]. However, the transformation is a greedy heuristic, which may not always produce the optimal solution. Another category of methods try to represent delay metrics in the abstract timing model with fewest edges. An optimal realization of a distance matrix problem is formulated as constructing a graph that preserves shortest-path distances while minimizing the total sum of edge weights [3], [4]. A clique-star replacement technique is proposed for the graph with unit edge delays [5], [6]. The clique is replaced by the star by 1) inserting a Steiner vertex at the center and 2) assigning 1/2 delay to each edge. However, if the graph has general edge delays, the clique may not be replaced by star due to infeasible edge delays. We are unaware of reports that can identify cliques with feasible edge delays for the star replacement in the graph with general edge delays.

In this paper, we propose a timing model reduction algorithm which minimizes the number of edges for timing propagations based on a biclique-star replacement technique. Our contributions are as follows.

- We derive a biclique-star replacement technique, which replaces a biclique of general edge delays by a star as far as the edge delays from various inputs share a common pattern. By inserting a Steiner vertex at the center of the biclique, we utilize the common pattern and cover multiple edge delays from each input by one edge, thus reducing the number of edges.
- We present a heuristic algorithm which searches the bicliques containing delay patterns. We allow don't-care edges in the delay pattern, thus maximizing the number of edges reduced.

The remainder of this paper is organized as follows. In section II, we introduce the terminologies. In section III, we introduce the biclique-star replacement technique. Section IV presents the timing model reduction algorithm. The experimental results are presented in Section V. Finally, we give the conclusions.

II. Terminology

The timing graph of a hierarchical block H is a weighted graph, denoted as G_H. The weight of each edge (i, j), denoted as edge delay $d_{i,j}$, is the corresponding gate or interconnect delay estimated based on the linear delay model. The delay of a path from input i to output j, denoted as $d_{p_{i,j}}$, is the total delay of edges on the path. The shortest path delay from input i to output j in G_H, denoted as $d^{min}_{H_{i,j}}$, is the minimum of all path delays $d_{p_{i,j}}$ in G_H. The longest path delay from input i to output j, denoted as $d^{max}_{H_{i,j}}$, is the maximum of all path delays $d_{p_{i,j}}$ in G_H.

The timing model of a hierarchical block H is a weighted graph G_M, which has the same input set B and output set D as timing graph G_H, and an edge set E. The shortest and longest path delays from input i to output j in G_M are equal to $d^{min}_{H_{i,j}}$ and $d^{max}_{H_{i,j}}$ in G_H. Note the internal vertices in timing model G_M and edges in edge set E may or may not be the same as those in timing graph G_H. A bipartite maximum delay model, denoted as G^{max}_M, is a timing model, in which any vertex is either an input or an output. On each edge (i,j) in G^{max}_M the attached edge delay $d_{i,j}$ is equal to the longest path delay $d^{max}_{H_{i,j}}$. A bipartite minimum delay model, denoted as G^{min}_M, is a timing model, in which any vertex is either an input or an output. On each edge (i,j) in G^{min}_M the attached edge delay $d_{i,j}$ is equal to the shortest path delay $d^{min}_{H_{i,j}}$.

Fig.1.(a) illustrates a timing graph G_H of a hierarchical block and the corresponding bipartite maximum delay model G^{max}_M. The input set contains three inputs, i.e., $B = \{1,2,3\}$. The output set contains three outputs, i.e., $D = \{9,10,11\}$. On edges between connected inputs and outputs the longest path delays are attached. For example, the longest path from input 1 to output 10 is $\{(1,4),(4,5),(5,7),(7,8),(8,10)\}$ which has delay 7. Thus, the delay attached on edge (1,10) is 7.

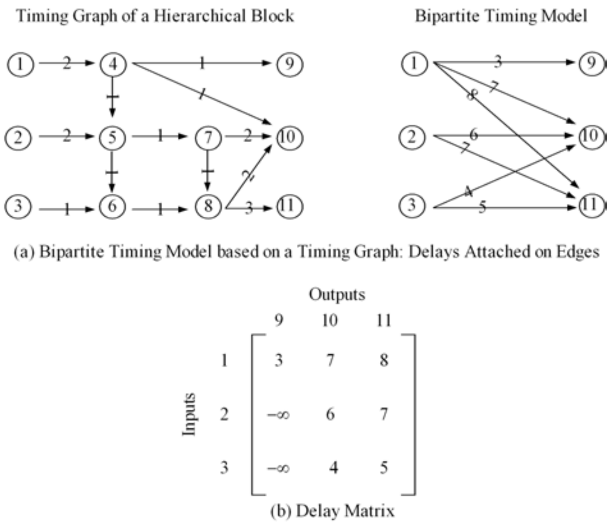
We formulate a delay matrix $M(G^{max}_M)$ based on the edge delays in bipartite maximum delay model G^{max}_M. The number of rows, denoted as r, is the number of inputs, i.e., $r = |B|$. The number of columns, denoted as c, is the number of outputs, $c = |D|$. The element on the ith row the jth column, denoted as $m_{i,j}$, is (1) edge delay $d_{i,j}$ if edge $(i,j) \in E$, or (2) $-\infty$ if input i disconnects with output j. The input delay vector of input i, denoted as I_i, is a set of elements on the ith row in matrix M, i.e., $I_i = \{m_{i,j} | j \in [1..c]\}$. The output delay vector of output j, denoted as O_j, is a set of elements on the jth column in matrix M,i.e., $O_j = \{m_{i,j} | i \in [1..r]\}$. Similarly, we can formulate the delay matrix for the bipartite minimum delay model. The fill-in in the minimum delay matrix for disconnected input-output is ∞.

The delay matrix of the bipartite maximum delay model in Fig.1.(a) is shown in Fig.1.(b). Each row in the matrix contains the edge delays from one input to all the connected outputs. For example, the first row contains edge delays from input 1 to outputs 9, 10, and 11, i.e., $d_{1,9} = 3$, $d_{1,10} = 7$ and $d_{1,11} = 8$. If the input is disconnected with an output, the delay is set to $-\infty$. For example, input 2 is disconnected with output 9, thus, the element on the 2nd row the 1st column is set to $-\infty$.

A biclique is a complete bipartite graph $G_c = \{B_c, D_c, E_c\}$, i.e., $\forall$ input i in input set B_c is connected with $\forall$ output j in output set D_c, i.e., $E_c = \{(i,j) | i \in B_c, j \in D_c\}$. Fig.2 illustrates a biclique and the corresponding delay matrix. Each pair of input and output is connected. In the delay matrix, there is no disconnected symbol $-\infty$.

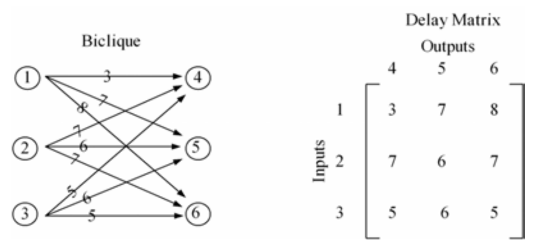
A star with a center vertex s is a weighted graph $G_s = \{B_s, D_s, s, E_s\}$, where B_s is the input set, D_s is the output set, s is the vertex at the center, and E_s is a set of edges from inputs to s and from s to outputs. Each edge has a weight. The weights of edges (i,s) and (s,j) are denoted as $d_{i,s}$ and $d_{s,j}$, respectively.

Fig.3 illustrates a star. The vertex s at the center connects with all inputs and outputs. On each edge, a weight is attached, i.e., edge delay $d_{i,s}$ or $d_{s,j}$. For example, edge delay $d_{1,s}$ of edge $(1,s)$ is 1.


Timing Graph of a Hierarchical Block
Bipartite Timing Model
(a) Bipartite Timing Model based on a Timing Graph: Delays Attached on Edges
Outputs
Inputs
(b) Delay Matrix

$$\begin{array}{c|ccc} & 9 & 10 & 11 \\ \hline 1 & 3 & 7 & 8 \\ 2 & -\infty & 6 & 7 \\ 3 & -\infty & 4 & 5 \end{array}$$

Fig. 1. Bipartite Timing Model and Delay Matrix



$$\begin{array}{c|ccc} & 4 & 5 & 6 \\ \hline 1 & 3 & 7 & 8 \\ 2 & 7 & 6 & 7 \\ 3 & 5 & 6 & 5 \end{array}$$

Fig. 2. Biclique and the Delay Matrix

III. Biclique-Star Replacement

In this section, we propose a biclique-star replacement technique which replaces bicliques with general edge delays by stars. Intuitively, a biclique with unit edge delays can be replaced by a star, such that the edge delays are covered with fewer edges. However, if a biclique contains general edge delays, the delays may not coincide to be covered by a star. We match edge delays from various inputs to a common delay pattern and construct the star based on the pattern. By doing so, we cover multiple edges from one input by one edge, thus reducing the number of edges.

A. Replacement Covering All Edge Delays

In this section, we propose to replace a biclique by a star and cover all the edge delays in the biclique by the star. We define the edge delay coverage and the biclique-star replacement first. After that, based on the observation on an example, we

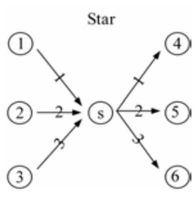


Fig. 3. Star

introduce the technique matching edge delays to a pattern and replacing a biclique by a star.

Definition 3.1: (**Edge Delay Coverage**) Edge (i,j) in biclique G_c is covered in a star G_s if $d_{i,s}+d_{s,j}=d_{i,j}$, where $d_{i,j}$, $d_{i,s}$ and $d_{s,j}$ are edge delays in G_c and G_s.

Definition 3.2: (**Biclique-star replacement**) A biclique-star replacement replaces biclique G_c by a star G_s such that

1) $B_s=B_c$, $D_s=D_c$, where B_s and D_s are input and output sets of G_s, B_c and D_c are input and output sets of G_s;
2) all edges in biclique G_c are covered in G_s.

The reduction ratio is the number of edges in G_c over the number of edges in G_s, i.e.,

$$r=(r\times c)/(r+c), \tag{1}$$

where r and c are the number of inputs and outputs in G_c.

One observation on biclique-star replacement is that the delays from various input delay vectors are a pattern plus various offsets. For example, Fig.4.(a) illustrates a biclique-star replacement. Each edge in the biclique is covered by a two-edge path in the star. For example, the edge (1,4) in the biclique is covered by the path $\{(1,s),(s,4)\}$ in the star because $d_{1,4}=d_{1,s}+d_{s,4}=2$. The delay matrix is shown in Fig.4.(b). We find out that delay vector of input 1 is $0+\{2,3,4\}$, delay vector of input 2 is $1+\{2,3,4\}$, and delay vector of input 3 is $2+\{2,3,4\}$. Thus, the delay vector $\{2,3,4\}$ is the common pattern shared by three input delay vectors.

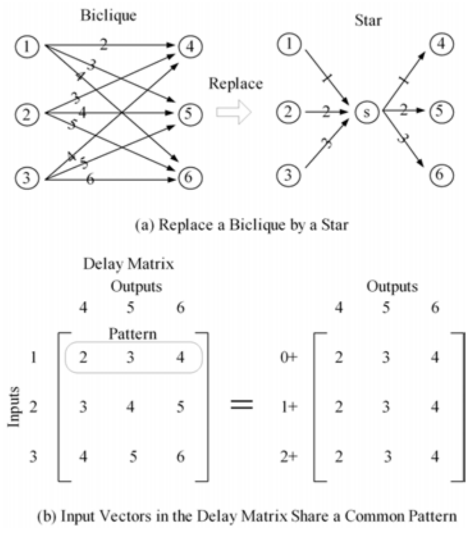

Biclique
Star
Replace
(a) Replace a Biclique by a Star
Delay Matrix
Outputs
Pattern
Inputs
(b) Input Vectors in the Delay Matrix Share a Common Pattern

Fig. 4. Biclique-Star Replacement

The hint of the example is that we can replace a biclique by a star as far as the input delay vectors share a common pattern. To identify the pattern in the input delay vectors, we define a vector subtraction operation as follows.

Definition 3.3: (**Vector Subtraction**) Given a delay matrix M of a biclique G_c, a vector subtraction between two input delay vectors I_a and I_b, denoted as $Sub(I_a,I_b)$, performs subtractions

$$\delta_j^{I_a,I_b}=m_{a,j}-m_{b,j}, \tag{2}$$

where $m_{a,j}\in I_a$ and $m_{b,j}\in I_b$ and returns a distance vector $V^{I_a,I_b}=\{\delta_j^{I_a,I_b}|j\in[1..c]\}$, where c is the number of column in M.

Definition 3.4: (**Pattern of input delay vectors**)

1. Two input delay vectors I_a and I_b in delay matrix of biclique G_c share a pattern vector, denoted as $\widehat{I_a,I_b}$, if in the distance vector V^{I_a,I_b} after vector subtraction $Sub(I_a,I_b)$, all $\delta_j^{I_a,I_b}$s in distance vector V^{I_a,I_b} are equal. The value is termed δ^{I_a,I_b}.
2. If $\forall$ two input delay vectors I_a and I_b in delay matrix M share a pattern vector, i.e., $\widehat{I_a,I_b}$, the biclique G_c contains an input pattern vector, denoted as $\widehat{G_c}$.

The biclique in Fig.4.(a) is an example of $\widehat{G_c}$. In the delay matrix in Fig.4.(b), input delay vectors $I_1=\{2,3,4\}$, $I_2=\{3,4,5\}$, and $I_3=\{4,5,6\}$. These three input delay vectors share a patten vector $\{1,1,1\}$.

Lemma 3.1: Given delay matrix M of biclique G_c, the complexity to evaluate the pattern vector in G_c is $O(r\times c)$, where r and c are the numbers of rows and columns in M.

If a biclique contains an input pattern vector, we can replace the biclique by a star using algorithm as follows.

```
Algorithm: Biclique-Star-Replacement($G_c$)
1) Star $G_s=\{B_s,D_s,s,E_s\}$, where input set $B_s=B_c$,
   output set $D_s=D_c$, and edge set $E_s=\{(i,s)|i\in B_s\}\cup\{(s,j)|j\in D_s\}$;
2) Pick input $0\in B_s$ and assign $d_{0,s}=0$, $d_{s,j}=d_{0,j}$ for
   edge $(s,j)\in E_s$;
3) For each input $i\in B_s$
       $d_{i,s}=d_{i,0}-d_{0,0}$, where $d_{i,0}$ and $d_{0,0}$ are delay of
       edges $(i,0)$ and $(0,0)$ in $G_c$;
```

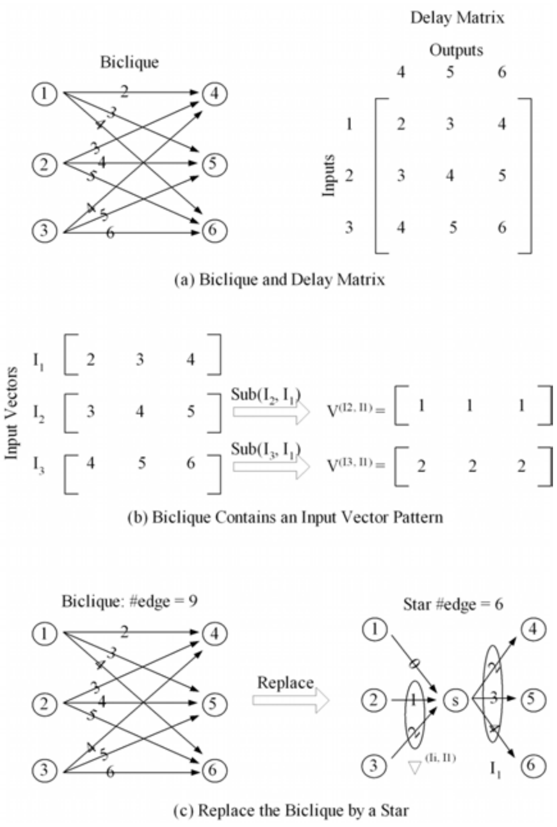
Fig.5.(a) illustrates a biclique and the delay matrix. In Fig.5.(b), we perform vector subtractions $Sub(I_2,I_1)$ and $Sub(I_3,I_1)$ and get the distance vectors $V^{I_2,I_1}=\{1,1,1\}$ and $V^{I_3,I_1}=\{2,2,2\}$. Since the δs in each distance vector are equal, the biclique contains an input pattern vector. In Fig.5.(c), we construct a star for the biclique. We first cover input delay vector I_1 by setting edge delays $d_{1,s}=0$, $d_{s,4}=d_{1,4}=2$, $d_{s,5}=d_{1,5}=3$, and $d_{s,6}=d_{1,6}=4$. Then, we cover input delay vectors I_2 and I_3 by setting edge delays $d_{2,s}=\delta^{I_2,I_1}=1$ and $d_{3,s}=\delta^{I_3,I_1}=2$. As a result, all the edges in the bicliques are covered in the star and the number of edge is reduced from 9 to 6.

B. Replacement Allowing Don't Care Edges

We allow don't-care edges thus generalizing biclique-star replacement to all bicliques. After the replacement, the edge delays in the biclique may or may not be covered in the star. However, as far as the number of edges covered is more than the number of edges used in the star, the replacement is beneficial.

A biclique can be replaced by star as far as the delay from an input to an output in the star does not dominate the corresponding edge delay in the biclique. The biclique-star replacement allowing don't-care edge is defined as follows.

Definition 3.5: (**Biclique-star Replacement Allowing Don't Care Edges**) Given a maximum delay biclique G_c, a



Biclique
Delay Matrix
Outputs
Inputs
(a) Biclique and Delay Matrix
Input Vectors
Sub(I2, I1)
Sub(I3, I1)
(b) Biclique Contains an Input Vector Pattern
Biclique: #edge = 9
Replace
Star #edge = 6
(c) Replace the Biclique by a Star

Fig. 5. Biclique-Star Replacement Based on Delay Pattern

biclique-star replacement allowing don't care edges replaces G_c by a star G_s such that

1) $B_s = B_c$, $D_s = D_c$, where B_s and D_s are input and output sets of G_s, B_c and D_c are input and output sets of G_c;
2) For edge $(i, j) \in E_c$, the delays $d_{i,s}$ and $d_{s,j}$ of edges (i, s) and (s, j) in E_s satisfy

$$d_{i,s} + d_{s,j} \le d_{i,j}. \tag{3}$$

Edge (i, j) is a don't care edge. The reduction ratio is the number of edges covered in G_s over the number of edges in G_s. Note an edge is covered in G_s only if $d_{i,s} + d_{s,j} = d_{i,j}$.

For a minimum delay biclique, the definition is similar except that the expression 3 is reversed in the direction of its inequality.

By allowing don't care edges, we can replace a biclique by a star when some sub-vectors of input delay vectors share a delay pattern and all other edges are don't care edges. After replacement, the edge delays in the sub-vectors sharing a pattern are covered. The sub-vectors sharing patterns and sub-vectors of don't care edges are defined as follows.

Definition 3.6: .

- **Sub-vector Sharing Pattern** Given input delay vectors I_a and I_b, the sub-vector $I_b^\delta \subseteq I_b$ shares a pattern with corresponding sub-vector $I_a^\delta \subseteq I_a$, i.e., $I_a^\delta = \{d_{a,j} | d_{b,j} \in I_b^\delta\}$, under δ if for $\forall d_{b,j} \in I_b^\delta$, $d_{b,j} - d_{a,j} = \delta$, i.e., $\forall \delta_j^{I_b,I_a} \in V^{I_b,I_a}$ equals δ, where V^{I_b,I_a} is the distance vector.
- **Sub-vector of Don't Care Delays** Given input delay vectors I_a and I_b, a delay $d_{b,j}$ is a don't care delay under δ if $\delta_j^{I_b,I_a} > \delta$. All the don't care delays formulates the sub-vector, termed $I_b^{\delta *}$.

An example of sub-vectors sharing pattern and sub-vectors of don't care delays are in Fig.6. Fig.6.(a) illustrates a biclique and the corresponding delay matrix. In Fig.6.(b), we perform vector subtraction $Sub(I_2, I_1)$ and $Sub(I_3, I_1)$. The distance vector $V^{I_3,I_1} = \{0, 0, 1, 1\}$. Under $\delta = 0$, the sub-vector $\{2, 3\} \in I_3$ shares pattern with I_1, and the sub-vector $\{5, 6\} \in I_3$ contains don't care delays. Under $\delta = 1$, the sub-vector $\{5, 6\} \in I_3$ shares pattern with I_1, and the sub-vector of don't care delays is empty.

When allowing don't care edges, $\forall$ biclique G_c can be replaced by a star G_s as follows.

Algorithm: Biclique-Star-Replacement-Allowing-Don't-Care(G_c,a)

```
1) Construct star $G_s = \{B_s, D_s, s, E_s\}$, where input set $B_s = B_c$, output set $D_s = D_c$, and edge set $E_s = \{(i, s)|i \in B_s\} \cup \{(s, j)|j \in D_s\}$;
2) Randomly choose input $a$ and assign $d_{a,s} = 0$, $d_{s,j} = d_{a,j}$ for each edge $(s, j) \in E_s$;
3) For each input $i \in B_s$
   a) Vector subtraction $Sub(I_a, I_i)$;
   b) $d_{i,s} = \min\{\delta_j^{I_a,I_i} | \delta_j^{I_a,I_i} \in V^{I_a,I_i}\}$;
```

We keep the direct edge (i, j) in the timing model to cover the edge delay of each don't care edge when replacing a biclique by a star. An example of the replacement allowing don't care edges is illustrated in Fig.6. Fig.6.(a) illustrates the biclique and the delay matrix. In Fig.6.(b), we get the distance vectors V^{I_2,I_1} and V^{I_3,I_1}. The minimum δs in V^{I_2,I_1} and V^{I_3,I_1} are 1 and 0, respectively. In Fig.6.(c), we construct the star by assigning delays in input delay vector I_1, i.e., $\{2, 3, 4, 5\}$, to edges $(s, 4)$, $(s, 5)$, $(s, 6)$ and $(s, 7)$. The minimum $\delta^{I_2,I_1} = 1$ and $\delta^{I_3,I_1} = 1$ are assigned to edges $(2, s)$ and $(3, s)$. We keep the don't care edges (3,6) and (3,7) after the replacement. From the biclique to the star, the number of edge is reduced from 12 to 9.

IV. Timing Model Reduction Based on Biclique-Star Replacements

In this section, we search bicliques containing delay patterns in the abstract timing model and minimize the number of edges by replacing bicliques by stars. The problem is equivalent to the minimum biclique covering problem without considering the edge delays, which is NP complete [6]–[8]. Therefore, we develop a set of heuristics to solve the problem in polynomial time.

A. Main Flow of Bipartite Timing Model Reduction

The main flow contains three steps. Firstly, we achieve a set of bicliques in the timing model as the replacement candidates. After that, we evaluate the reduction ratio for each biclique. A high reduction ratio indicates that a large number of edges can be reduced after the replacement. Finally, we choose the biclique with the maximum reduction ratio to replace.

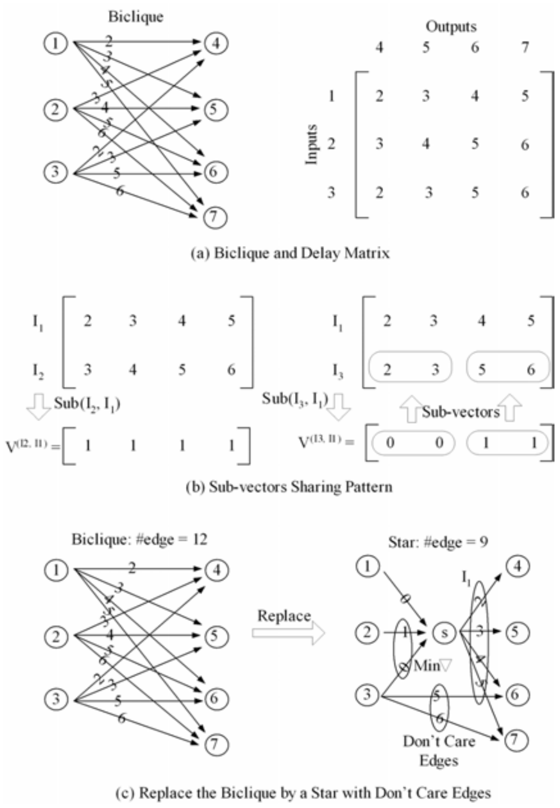


Fig. 6. Biclique-Star Replacement Allowing Don't Care Edges

Bipartite Timing Model Reduction(G)

```
1) Biclique Pool = Biclique-Search(G);
2) Repeat
    a) Evaluate the reduction ratio for each biclique in Biclique Pool;
    b) if max reduction $> 1$
        i) Replaces $G_c$ with the max reduction by a star;
        ii) Remove $G_c$ from BicliquePool;
3) Until max reduction $< 1$
```

The Indentify-Bicliques procedure returns a set of bicliques as the candidates to be replaced. We evaluate all the bicliques in the Biclique Pool, and replace the one with the maximum reduction ratio by a star using the Biclique-Star-Replacement-Allowing-Don't-Care algorithm. We repeat the evaluation and replacement steps until all the reduction ratios are smaller larger than 1, which means the number of edges can not be reduced further.

B. Biclique Search in Bipartite Timing Model

We search bicliques in the timing model as the replacement candidates and try to maximize the edge reduction. Although any biclique can be replaced by a star by allowing don't care edges, the edge reduction produced by the replacement is different. We devise two rules for biclique search according to reduction ratio defined in equation 1. The definition is not accurate for some cases, such as the biclique including don't care edges and some edges covered in various bicliques. However, it indicates the reduction potential of the biclique, and thus can be used in the biclique search.

1) *Maximize Biclique Size* Bicliques of larger size potentially have higher reduction ratio.
2) *Maximize Edge Coverage* We try to cover as many as possible edges with bicliques. If an edge is not covered by any biclique, we need one direct edge to cover the edge delay in the timing model after minimization. However, if the edge is covered by a biclique with reduction ratio r, after the biclique is replaced by a star, the edge delay is covered by $1/r$ edge. As far as $r > 1$, there are benefits.

Following these two rules, the biclique search algorithm is as follows.

Algorithm: Biclique-Search(G)

```
1) Biclique Pool = $\varnothing$;
2) Repeat
    a) Randomly choose edge $(p, q) \in E$ which is not covered by any biclique;
    b) Input set $B_c = \{p\}$, edge set $E_c = \{(p, j)|(p, j) \in E\}$, output set $D_c = \{j|(p, j) \in E\}$;
    c) For each input $i$ connected with output $q$ Biclique-Expansion$(G, G_c, p, q, i)$;
    d) Add biclique $G_c$ to Biclique Pool;
3) Until all edges covered;
```

In the algorithm, we iteratively construct bicliques starting from uncovered edges thus maximizing the edge coverage. When we construct the biclique for edge (p, q), all the edges (p, j) from input p are added to the biclique first. Then, we expand the biclique to cover edges from other inputs. When performing biclique expansion to input i, we try to cover as many as possible edges from input i and remove as few as possible edges already in the biclique. By doing so, the size of the biclique is maximized. The Biclique-Expansion algorithm is as follows.

Algorithm: Biclique-Expansion(G, G_c, p, q, i)

```
1) Vector subtraction $Sub(I_i, I_p)$ between input delay vectors $I_i$ and $I_p$;
2) max = 0;
3) For each $\delta \leq \delta_q^{I_i, I_p}$
    a) Get sub-vector $I_i^\delta$ sharing a pattern with $I_p$ under $\delta$;
    b) Get sub-vector $I_i^{\delta *}$ of don't care delays under $\delta$;
    c) current = Added-over-Removed($G_c, I_i^\delta, I_i^{\delta *}, \delta$);
    d) If (current $>$ max)
       max = current, max vector = $I_i^\delta$;
4) If max $> 0$
    a) For each output $j$ in $D_c$
        i) If $d_{i,j} \in I_i^\delta$ Add edge $(i, j)$ to $E_c$;
        ii) else Remove output $j$ and all input edges to $j$ from $G_c$;
```

Function: Added-over-Removed$(G_c, I_i^\delta, I_i^{\delta *}, \delta)$

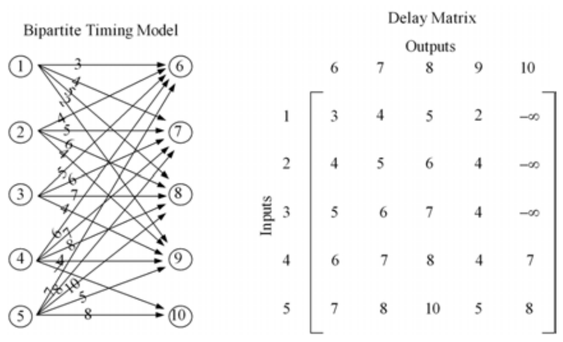
```
1) Added = $|I_i^\delta|$, Removed = 0;
2) For each output $j$ in $D_c$
    a) If $d_{i,j} \notin I_i^\delta \cup I_i^{\delta *}$
```

b) Removed = Removed + #Edges to output j with delays covered in G_c;

3) Return(Added - Removed);

We first perform vector subtraction between input delay vectors I_i and I_p. Then, we use the Added-over-Removed function to evaluate the number of edges covered for each $\delta \leq \delta_q^{I_i,I_p}$ as if we perform the Biclique-Star-Replacement with $d_{i,s} = \delta$. According to the Biclique-Star-Replacement-Allowing-Don't-Care algorithm, when assigning $d_{i,s} = \delta$, an edge (i,j) is covered if delay $d_{i,j} \in I_i^\delta$, where I_i^δ is a sub-vector of input delay vector I_i sharing a pattern with I_p. The edges (i,j) with delays $d_{i,j} \notin I_i^\delta \cup I_i^{\delta *}$ can not be added. Therefore, we need to remove output j to keep the biclique complete. As a result, the edges to j which are originally covered in the biclique are counted as removed edges. After the evaluation, we expand the biclique based on the δ which maximizes the edge coverage. The edge (i,j) is added if $d_{i,j}$ belongs to the union $I_i^\delta \cup I_i^{\delta *}$. Otherwise, the output j and the edges to j are removed. By restricting $\delta_j^{I_i,I_p} \leq \delta_q^{I_i,I_p}$, we ensure that $d_{i,q} \in I_i^\delta \cup I_i^{\delta *}$, thus keeping output q and edge (p,q) in the biclique.

Fig.8 and Fig.9 illustrate a biclique expansion example based on the bipartite timing model in Fig.7. We want to construct the maximum biclique covering edge (1,6). Fig.8 illustrates steps 1, 2 and 3 of the expansion. Fig.9 illustrates steps 4 and 5.



$$\begin{array}{c|ccccc} & 6 & 7 & 8 & 9 & 10 \\ \hline 1 & 3 & 4 & 5 & 2 & -\infty \\ 2 & 4 & 5 & 6 & 4 & -\infty \\ 3 & 5 & 6 & 7 & 4 & -\infty \\ 4 & 6 & 7 & 8 & 4 & 7 \\ 5 & 7 & 8 & 10 & 5 & 8 \end{array}$$

Fig. 7. Bipartite Timing Model and the Delay Matrix Example

C. Edge Reduction Evaluation

We use reduction ratios to evaluate benefits of replacing bicliques by stars. However, the reduction ratio defined in equation 1 is not accurate when there are don't care edges in the biclique or some edges are covered by multiple bicliques. A more accurate and general definition of the reduction ratio is as follows.

Definition 4.1: (**Reduction Ratio**) Given a biclique G_c, if G_c can be replaced by a star G_s, the reduction ratio $r = c/(m+n)$, where c is the number of edge delays covered and only covered by G_s, $m+n$ is the number edges in G_s.

Therefore, when replacing a biclique by a star, we label the edges covered by the star. After the replacement, we re-compute the reduction ratios for the bicliques left in the Biclique Pool. All the edges with labels are not counted.

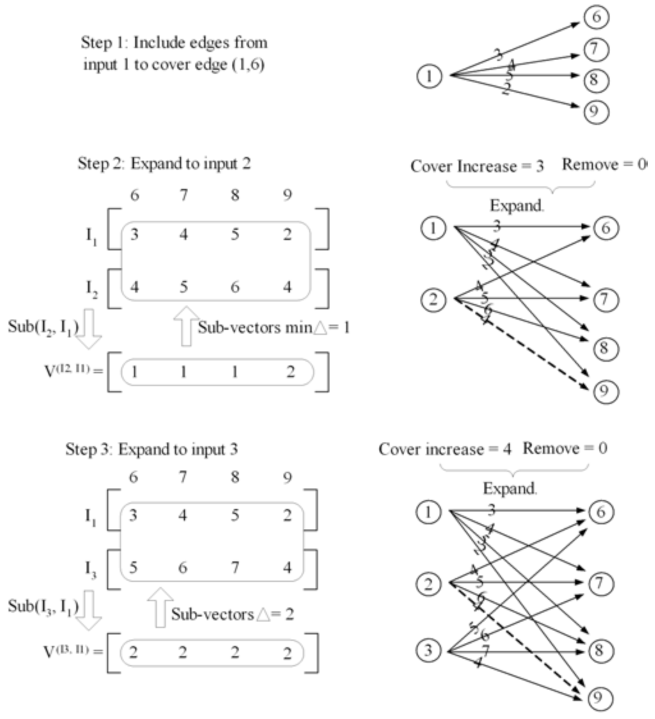

Step 1: Include edges from input 1 to cover edge (1,6)
Step 2: Expand to input 2
Cover Increase = 3 Remove = 0
Expand.
Sub(I2, I1)
Sub-vectors min△= 1
Step 3: Expand to input 3
Cover increase = 4 Remove = 0
Expand.
Sub(I3, I1)
Sub-vectors△= 2

Fig. 8. Biclique Expansion Starting from Edge (1,6) in Bipartite Timing Model (Fig.7): Steps 1 to 3.

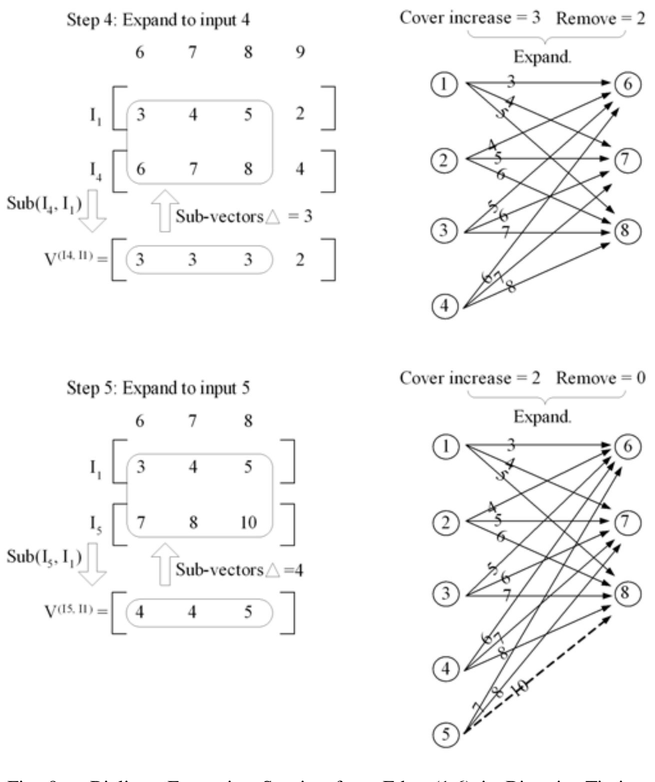

Step 4: Expand to input 4
Cover increase = 3 Remove = 2
Expand.
Sub(I4, I1)
Sub-vectors△ = 3
Step 5: Expand to input 5
Cover increase = 2 Remove = 0
Expand.
Sub(I5, I1)
Sub-vectors△ =4

Fig. 9. Biclique Expansion Starting from Edge (1,6) in Bipartite Timing Model (Fig.7) : Steps 4 and 5.

We repeat the reduction ratio evaluation and biclique-star replacement steps until no bicliques in the biclique pool can provide further edge reduction. The timing model after reduction is shown in Fig.10. The number of edges in the bipartite timing model is reduced from 22 to 16.

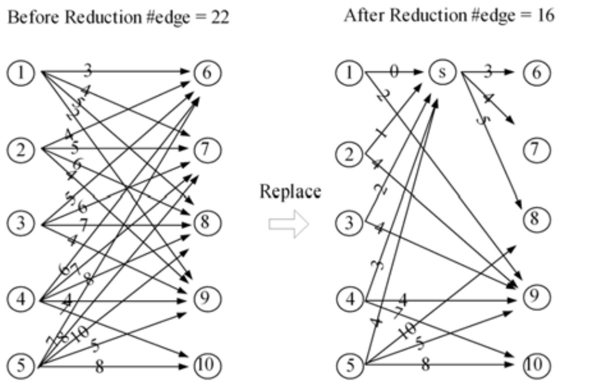


Fig. 10. Bipartite Timing Model (Fig.7) Reduction : The number of edges is Reduced from 22 to 16.

***Theorem** 4.1:* The complexity of constructing maximum biclique for edge (p,q) is $O_{pq} = O((d^-(q) \times d^+(p)^2))$, where $d^+(p)$ and $d^-(q)$ are output and input degree of input p and output q. The complexity of biclique evaluation $O_e = O(|E| \times k)$, where E is the edge set of the timing model and k is the number of bicliques in Biclique Pool. The complexity of timing model reduction is $O(\Sigma(O_{pq}) + k \times O_e)$.

D. Iterative Timing Model Reduction

We iteratively perform the bipartite timing model reduction to reduce the number of edges further. A vertex splitting and a star recover technique are proposed to maintain the timing model a bipartite graph. Based on the bipartite timing model, we can repeat the bipartite timing model reduction process until no improvement, thus maximizing the edge reduction.

After replacing a set of bicliques by stars, the timing model is not bipartite graph any more. The inserted vertices, which are the centers of the stars, partition a part of the timing model into two bipartite graphs. Intuitively, we can repeat the timing model reduction on each bipartite partition and accumulate the results into the whole timing model. However, we may lose the ability to discover larger bicliques crossing multiple bipartite partitions, which makes the reduction less efficient. For example in Fig.11, the timing model is partitioned by vertices s_1 and s_2 into several bipartite graphs. The biclique circled in the figure which includes inputs 1,2,3, vertices s_1 and s_2, and output 9, is hard to be discovered.

We propose a vertex splitting technique to transform the timing model into a bipartite graph. By doing so, we can discover larger bicliques thus improving the reduction ratio. We split vertex s of each star G_s as follows.

Algorithm: Vertex-Splitting(G, G_s)

```
1) Split vertex $s$ into vertices $s$ and $s'$;
2) Input set $B = B \cup \{s'\}$, output set $D = D \cup \{s\}$;
3) For each output $j$ in output set $D_s$
      $E = E - \{(s,j)\}$ and $E = E \cup \{(s',j)\}$
```

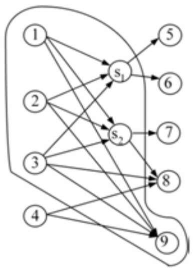


Fig. 11. Bicliques Crossing Multiple Bipartite Partitions

We add the duplicated vertex s' into the input set and push vertex s into output set D. All the edges originally from s to j are moved to vertex s'.

As the reverse process of the vertex splitting, we recover a star by merging the corresponding vertices s' and s.

Algorithm: Star-Recover(G', s, s')

```
1) $B = B - \{s'\}$, $D = D - \{s\}$;
2) For each edge $(s',j)$
      Add edge $(s,j)$;
```

Fig.12 illustrates an example of the vertex splitting and star recover. From left to right, we split vertices s_1 and s_2, add s'_1 and s'_2 into the input set, and push s_1 and s_2 into the output set. After the vertex splitting, the timing model G is transformed into a bipartite graph. The star recover is the reverse process, which merges s_1 with s'_1, and s_2 with s'_2. The bipartite graph is recovered into the timing model.

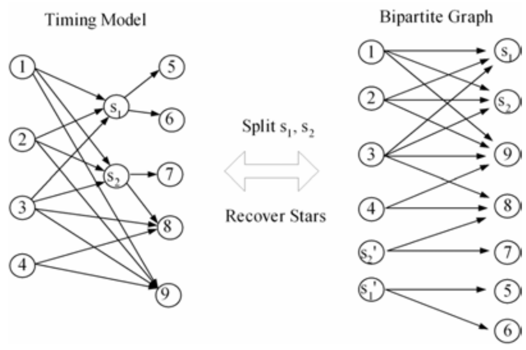


Fig. 12. Vertex Splitting and Star Recover

With the vertex splitting and star recover, we can iteratively perform the bipartite timing model reduction to minimize the number of edges in the timing model.

Algorithm: Iterative-Reduction(G)

```
1) Repeat
      a) Bipartite-Timing-Model-Reduction($G$);
      b) For all stars $G_s$ Vertex-Splitting($G, G_s$)
2) Until no edge reduction
3) For all duplicate vertices $s$ and $s'$ Star-Recover($G, s, s'$)
```

***Theorem** 4.2:* Edge delay $d_{i,j}$ of any connected input i and output j in timing model G is covered by the longest path delay $d'_{i,j}$ from input i to output j in timing model G' after the reduction.

TABLE I

EDGE REDUCTION WITH ERROR BOUNDS

Block 1 E_G = 138,360 E_B = 262,491				Block 2 E_G = 103,414 E_B= 465,190			
Error(ns)	E_a	r_G	r_B	Error(ns)	E_a	r_G	r_B
0	249,032	-80.0%	5.1%	0	397,384	-284.3%	14.6%
0.1	41,696	69.9%	84.1%	0.01	49,613	52.0%	89.3%
1.0	36,980	73.3%	85.9%	0.10	29,477	71.5%	93.7%
10.0	35,981	74.0%	86.3%	1.0	21,192	79.5%	95.4%
100.0	36,169	73.9%	86.2%	10.0	20,262	80.4%	95.6%
Buffer $\times$ 1 delay = 1.34ns.				Buffer $\times$ 1 delay = 0.74ns			

V. EXPERIMENTAL RESULTS

We test the proposed approach on industry test cases. The algorithm is implemented in C and run on a Pentium 4 Linux machine. We construct and minimize the timing models for two circuit blocks (Table I). Circuit block 1 contains 8499 inputs, 16885 outputs, 138,360 edges in the timing graph of the block, and 262,491 edges in the bipartite timing model. Circuit block 2 contains 4260 inputs, 103,414 edges in the timing graph of the block, and 7728 edges in the bipartite timing model.

We compare the number of edges in the timing model after reduction with both the number of edges in the timing graph of the block and the number of edges in the bipartite timing model. Two reduction ratios are defined as follows.

$$r_G = (E_G - E_m)/E_G \quad (4)$$
$$r_B = (E_B - E_m)/E_B, \quad (5)$$

where E_G is the number of edges in the timing graph of the block, E_B is the number of edges in the bipartite timing model, and E_m is the number of edges in the timing model after the reduction.

We allow error bounds on edge delays. For any connected input i and output j, the error bound is defined as follows.

$$d_{i,j} - d'_{i,j} \leq error\_bound, \quad (6)$$

where $d_{i,j}$ and $d'_{i,j}$ are the longest path delays in timing models before and after the model reduction. In Table I, we show the number of edges in the timing model after the reduction, i.e., E_m, and the reduction ratios, i.e., r_G and r_B. For block 1, if the error bound is 0, E_m will be larger than E_G and smaller than E_B because the number of edges is increased when we transform the timing graph of the block into the bipartite timing model. By allowing 0.1ns error bound, the number of edges is reduced by 69.9% compared with the timing graph and 84.1% compared with bipartite timing model.

We produce timing models of acceptable accuracy using small error bounds. In block 1, the delay of minimum size buffer, $Buffer \times 1$, is 1.34ns, which is the delay of a minimal sized inverter with one fan-out. Typically, the critical path delay is 10 to 20 times of the minimum buffer delay. Therefore, the impact of 0.1ns error bound on the critical path delay is less than 1%, which is acceptable in terms of the accuracy. By further increasing the error bounds, we can reduce more edges. However, the improvement is not substantial and if the error bound is too large, the abstract timing model is not accurate.

VI. CONCLUSION

We propose an abstract timing model reduction algorithm for hierarchical timing analysis based on a biclique-star replacement technique. The number of edges in the abstract timing model for timing propagations are minimized, thus improving the analysis efficiency. By allowing reasonable error bounds, the experiments results show that the proposed algorithm effectively reduces the number of edges in the timing model.

VII. ACKNOWLEDGEMENTS

This work was supported in part by the California MICRO program.

REFERENCES

[1] C. Visweswariah and A. R. Conn. Formulation of static circuit optimization with reduced size, degeneracy and redundancy by timing graph manipulation. In *Proc. of the Intl. Conf. on Computer-Aided Design*, page 244C251, 1999.

[2] C.W. Moon, H. Kriplani, and K. P. Belkhale. Timing model extraction of hierarchical blocks by graph reduction. In *Proc. of the Design Automation Conf.*, page 152C157, 2002.

[3] S. L. Hakimi and S. S. Yau. Distance matrix of a graph and its realizability. *Quart. Appl. Math.*, 22:305C317, 1964.

[4] F. Chung, M. Garrett, R. Graham, and D. Shallcross. Distance realization problems with applications to internet tomography. http://www.math.ucsd.edu/?fan.

[5] T. Feder, A. Meyerson, R. Motwani, L. OCallaghan, and R. Panigrahy. Representing graph metrics with fewest edges. In *Proc. of Symp. on Theoretical Aspects of Computer Science*, pages 355–366, 2003.

[6] T. Feder and R. Motwani. Clique partitions, graph compression and speeding up algorithms. In *Proc. of the ACM Symposium on Theory of Computing*, pages 123–133, 1991.

[7] J. Orlin. Containment in graph theory: Covering graphs with cliques. *Indag. Math.*, 39:211–218, 1977.

[8] H. Muller. On edge perfectness and classes of bipartite graphs. *Discrete Math.*, 149:159–187, 1996.