Performance-Driven Concurrent Placement and Gate Sizing for Deep Submicron Circuits

Wei Chen and Massoud Pedram
Department of EE – Systems
University of Southern California

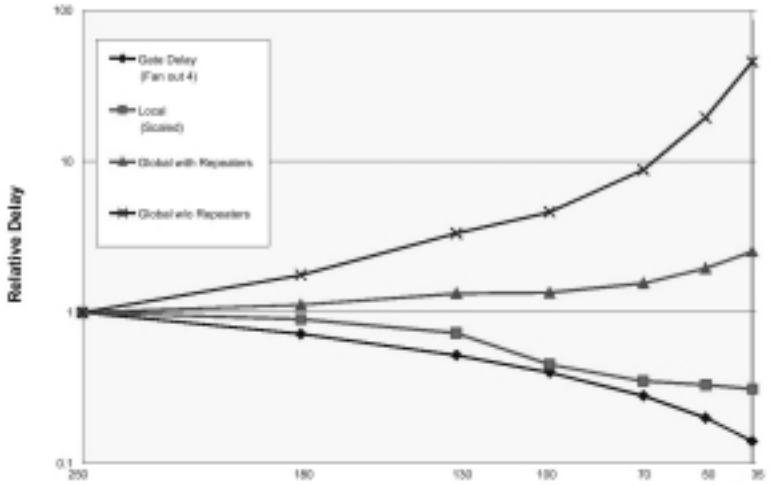
Outline

- Introduction
- Background
- Algorithm
- Other Work
- Future Directions

Introduction (Global View)

- Traditional design flows ignore the contribution of interconnects to overall circuit delay
- What is happening?
  - Smaller feature sizes (transistors)
  - More complex designs
  - Faster systems
- Interconnect delay can no longer be ignored

Interconnect Delay (ITRS 99)



Effects of Placement and Gate Sizing

- Placement
  - Assign the cells to suitable locations
  - Optimize the interconnect delay
  - A poorly placed layout cannot be improved by a subsequent high quality routing
- Gate Sizing
  - Tune the size of each cell
  - Optimize the gate delay
  - Balance the path delays in the circuit

Sequential Approach

- Flow:
  - Performance-driven placement followed by in-place gate sizing
- Pros:
  - Fast (need to solve two smaller independent problems)
- Cons:
  - In-place gate sizing ignores the delay optimization opportunity with respect to interconnect
  - No interaction between the two steps

Unification-based Approach

- Flow
  - Concurrent performance-driven placement and gate sizing
- Pros
  - Optimizes both interconnect delay and gate delay at the same time
  - Tight interaction between the two steps
- Cons
  - Higher complexity (need to solve a much larger problem)

Previous Works

- Placement
  - Minimize wire length
    - TimberWolf(Sechen '85),GORDIAN(Kleinhans '91)
  - Improve performance
    - Net-based: SPEED (Riess '95)
    - Path-based: RITUAL (Srinivasan '91)

Previous Works

- Gate Sizing
  - Discrete sizing
    - Coudert ’96
  - Continuous sizing
    - Berkelaar ’90
- Concurrent Re-location and Gate Sizing
  - Piecewise linear formulation
    - Chuang ’94

Outline

- Introduction
- Background
- Algorithm
- Other Work
- Future Directions

Placement Problem

- Given a collection of cells with ports on the boundaries, the dimensions of these cells, and a collection of nets, the process of placement consists of finding suitable physical locations for each cell
- NP-complete
- Traditional objective – minimize wire length
- More recent objective – improve performance

RITUAL – Problem Formulation

minimize: $L(w)$

s.t.

$$a_j \geq a_i + d(v_i, v_j) \qquad \forall (v_i, v_j) \in A$$

$$a_j \leq \mathrm{T_j} \qquad \forall v_j \in \mathbf{PO}$$

$$a_j \geq \mathrm{T_j} \qquad \forall v_j \in \mathbf{PI}$$

$$d(v_i, v_j) = f(X_i, Y_i) \qquad \forall n_i \in N$$

$$\frac{1}{|S_j|} \sum_{i \in S_j} x_i = r_j^x \qquad j = 1, \ldots, k$$

$$\frac{1}{|S_j|} \sum_{i \in S_j} y_i = r_j^y \qquad j = 1, \ldots, k$$

RITUAL – Lagrangian Relaxation

- Convex formulation
- Transform the original problem formulation to an unconstrained optimization problem by using the Lagrangian multipliers
- For any fixed value of Lagrangian multipliers, the unconstrained problem has a simple solution

Lagrangian Relaxation

$$\textbf{minimize} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{Q}\boldsymbol{w} + \boldsymbol{b}^T\boldsymbol{w}$$

$$\textbf{s.t.} \quad \boldsymbol{A}\boldsymbol{w} \le \boldsymbol{c}$$

$$\Downarrow$$

$$\max_{\lambda \ge \mathbf{0}} \left(\min_{x} \left(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{Q}\boldsymbol{w} + \boldsymbol{b}^T\boldsymbol{w} + \lambda^T(\boldsymbol{A}\boldsymbol{w} - \boldsymbol{c}) \right) \right)$$

$$\Downarrow$$

$$\boldsymbol{w}^{(k+1)} = -\boldsymbol{Q}^{-1}[\lambda^{(k)}\boldsymbol{A} + \boldsymbol{b}]$$

RITUAL – Reducing The Problem Size

- The above problem formulation becomes too large if we include all the cells in the circuit
- Reduced active forest (RAF): the set of paths connecting to the POs that violate timing requirements
- The problem formulation only contains timing constraints for RAF

Gate Sizing Problem

- Tune the gate sizes to improve the critical path delay
- Discrete gate sizing:
  - NP-complete
- Continuous gate sizing
  - Easy to solve

Berkelaar's Algorithm

- Path-based, similar formulation as RITUAL
- Continuous gate sizing model
  - Non-linear

$$d_{gate} = d_{int} + c \cdot \frac{C_{load}}{z_{gate}}$$

$$C_{load} = c_{wire} + \sum_{i \in fanout(gate)} z_i \cdot C_{in,i}$$

  - Piecewise linear

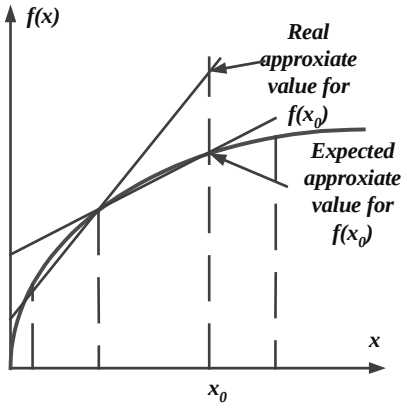
$$d_{gate} \geq c_{1,1} - c_{1,2} \cdot z_{gate} + c_{1,3} \cdot \sum_i z_i \cdot C_{in,i}$$

$$\cdots\cdots$$

$$d_{gate} \geq c_{n,1} - c_{n,2} \cdot z_{gate} + c_{n,3} \cdot \sum_i z_i \cdot C_{in,i}$$

Error in The Linear Approximation

- Non-convex delay function



Concurrent Relocation and Sizing (Chuang '94)

- Non-convex delay model

$$d_i = \frac{c}{z_i} \cdot (C_{wire} + C_{gate})$$

$$C_{wire} = C_h(x_{max} - x_{min}) + C_v(y_{max} - y_{min})$$

$$C_{gate} = \sum_{j \in fanout(i)} (\alpha_{i,j} \cdot z_j + \beta_{i,j})$$

- Piecewise linearlization of the delay equation
- Accuracy concerns

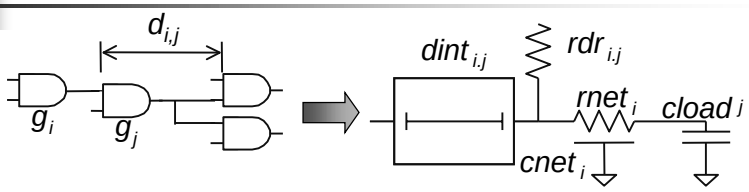
Outline

- Introduction
- Background
- Algorithm
- Other Work
- Future Directions

Our Problem Definition

- Given a mapped and placed circuit with the allowed range of gate sizes, find the best location and size for each gate in the circuit so as to minimize the circuit delay

Our Delay Model



$$d_{i,j} = dint_{i,j} + rdr_{i,j} \cdot (cload_j + cnet_j) + rnet_j \cdot cload_j$$

$$\text{where} \quad cload_j = \sum_{g_k \in fanout(g_j)} cin_{j,k}$$

Delay Model (Cont'd)

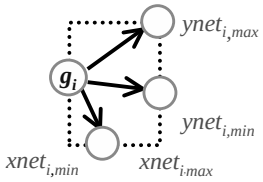
- Gate Sizing Model

$$dint_{i,j}(z_j) = \alpha 1_{i,j} \cdot z_j + \beta 1_{i,j}$$

$$rdr_{i,j}(z_j) = \frac{\alpha 2_{i,j}}{z_j} + \beta 2_{i,j}$$

$$cin_{i,j}(z_j) = \alpha 3_{i,j} \cdot z_j + \beta 3_{i,j}$$

- Wire Load Estimation



$$cnet_i = \rho \cdot [C_{hor}(xnet_{i,max} - xnet_{i,min}) + C_{ver}(ynet_{i,max} - ynet_{i,min})]$$

$$rnet_i = \rho \cdot [R_{hor}(xnet_{i,max} - xnet_{i,min}) + R_{ver}(ynet_{i,max} - ynet_{i,min})]$$

The Unified Delay Model

- Pin-dependent delay model
  - Non-convex

$$d_{i,j} = dint_{i,j}(z_j) + rdr_{i,j}(z_j) \cdot [\rho \cdot C_{hor}(xnet_{j,max} - xnet_{j,min}) + \rho \cdot C_{ver}(ynet_{j,max} - ynet_{j,min}) + \sum_{g_k \in fanout(g_j)} cin_{j,k}(z_k)]$$

$$+ \rho \cdot [R_{hor}(xnet_{j,max} - xnet_{j,min}) + R_{ver}(ynet_{j,max} - ynet_{j,min})] \cdot \sum_{g_k \in fanout(g_j)} cin_{j,k}(z_k)$$

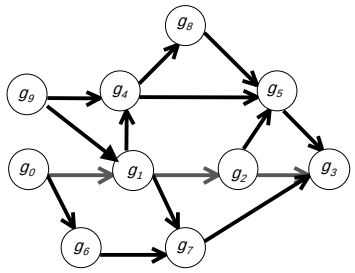
Why avoid Bisectioning

- Traditional placement based on mathematical programming resort to recursive bisectioning to achieve uniform cell distribution and/or improve path delay
- Problems
  - Unfixed gate size: cannot keep the partition balance
- Method
  - Set variable change region dynamically

Reduce Problem Size

- High problem complexity
  - Non-convex delay model
  - Numbers of variables and constraints increase with circuit size
- Method
  - Iteratively identify and optimize the critical sections

Motivational Example



- Critical path: $C(1)=\{g_0, g_1, g_2, g_3\}$
- Neighbor $(1,1)$: $Ne(1,1)=\{g_4, g_5, g_6, g_7\}$
- Critical section: $C(k) \cup Ne(k,1)$

Global Problem Formulation

$$\textbf{minimize} \quad t_{cycle}$$

$$\begin{aligned}
\textbf{s.t.} \quad & a_j \geq a_i + d_{i,j} && \forall (v_i, v_j) \in A \\
& a_j \leq \mathrm{T}_{\mathrm{start}} + t_{cycle} && \forall v_j \in \mathbf{PO} \\
& a_j \geq \mathrm{T}_{\mathrm{start}} && \forall v_j \in \mathbf{PI} \\
& \hat{x}_i^- \leq x_i \leq \hat{x}_i^+ && \forall v_i \in C(k) \\
& \hat{y}_i^- \leq y_i \leq \hat{y}_i^+ && \forall v_i \in C(k) \\
& \hat{z}_i^- \leq z_i \leq \hat{z}_i^+ && \forall v_i \in C(k)
\end{aligned}$$

Global Problem Formulation (Cont'd)

- It includes complete timing relations throughout the circuit
- It is too complicate for large circuits
- If variable change regions can be set correctly so as to guarantee that the changes in *C(k)* will not increase the delay of any path outside of *C(k)* beyond that of the current most critical path, the arrival time variables of the cells outside *C(k)* can be dropped from formulation

Critical Path Sizing & Placement

$$\textbf{minimize} \quad t_{cycle}$$

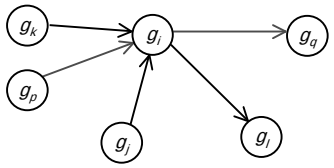
$$\begin{aligned}
\textbf{s.t.} \quad & a_j \geq a_i + d_{i,j} && \forall (v_i, v_j) \in A,\ v_i, v_j \in C(k) \\
& a_j \leq \mathrm{T}_{\mathrm{start}} + t_{cycle} && \forall v_j \in PO \text{ and } v_j \in C(k) \\
& a_j \geq \mathrm{T}_{\mathrm{start}} && \forall v_j \in PI \text{ and } v_j \in C(k) \\
& \hat{x}_i^- \leq x_i \leq \hat{x}_i^+ && \forall v_i \in C(k) \\
& \hat{y}_i^- \leq y_i \leq \hat{y}_i^+ && \forall v_i \in C(k) \\
& \hat{z}_i^- \leq z_i \leq \hat{z}_i^+ && \forall v_i \in C(k)
\end{aligned}$$

Dynamic Variable Change Regions (DVCRs)

- Rationale
  - Solution oscillation
  - Congestion consideration
- How to calculate the DVCRs
  - Determined by the slack time of the fan-ins and fan-outs
  - Reduced in size as the optimization progresses
- Three cases
  - Only one gate is repositioned
  - Only one gate is resized
  - All the critical cells can be resized and relocated

Only Gate g_i Is Repositioned

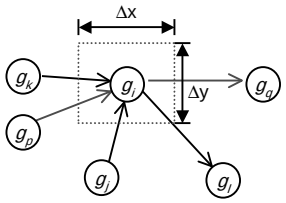
- To determine: $\hat{x}_i^-, \hat{y}_i^-, \hat{x}_i^+, \hat{y}_i^+$



- g_p g_q: critical fan-in and fan-out
- g_k g_j g_l: non-critical fan-ins and fan-outs

DVCR Calculation (Cont'd)

- DVCR induced by fan-in g_k

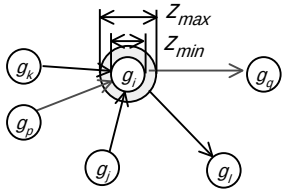


From the delay equation, calculate bounding box Δx, Δy such that the slack of g_k will always be greater than the current critical slack if g_i is placed inside the box

DVCR Calculation (Cont'd)

- DVCR induced by fan-out g_j and g_l
  - Similar to fan-in
- DVCR of g_i
  - Intersection of all the fan-in and fan-out induced DVCRs

Only Gate g_i Is Resized

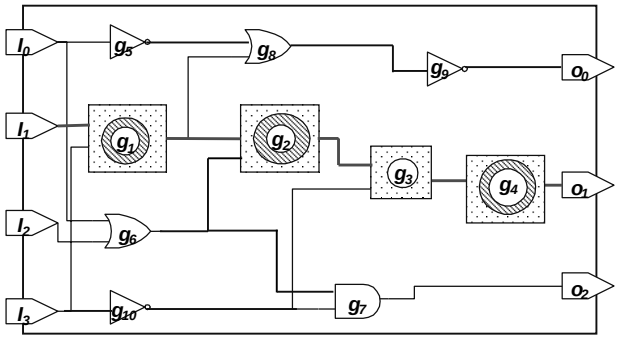


- From the delay equation, fan-in g_k determines the upper-bound z_{max} so that the slack of g_k will always be greater than the current critical slack
- Fan-outs g_j and g_l determine the lower-bound z_{min} similarly

All The Critical Cells Can Be Resized and Relocated

- Re-convergent problem
  - Fan-ins and Fan-outs of *C(k)* share the slack time of some common path
  - User-defined parameter μ ($0<\mu<1$) to scale down the DVCR calculated from the above
  - μ is decreased gradually
- Perturbation problem
  - Maximum location change value
  - The smaller of the location DVCR and the above maximum location change value

DVCR Example



- Max size: fan-in slack determined
- Min size: fan-out slack determined
- Location: fan-in & fan-out slack determined

Ne(k,1) Optimization

- Optimization Methods
  - Placement of the immediate fan-outs of the critical paths
  - Resizing of the immediate fan-outs of the critical paths
- Concerns
  - Good to do placement and resizing for *Ne(k,1)* at the same time
  - Too difficult to control the size of *Ne(k,1)*

Neighbor Repositioning

- Linear programming (LP)

minimize t_{cycle}

s.t.

$$a_j \geq a_i + d_{i,j} \qquad \forall (v_i, v_j) \in A$$

$$a_j \leq \mathrm{T}_{\mathrm{start}} + t_{cycle} \qquad \forall v_j \in PO$$

$$a_j \geq \mathrm{T}_{\mathrm{start}} \qquad \forall v_j \in PI$$

$$| x_i - \hat{x}_i | \leq \Delta x \qquad \forall v_i \in Ne(k,1)$$

$$| y_i - \hat{y}_i | \leq \Delta y \qquad \forall v_i \in Ne(k,1)$$

Neighbor Resizing

- Geometric Programming

minimize t_{cycle}

s.t.

$$a_j \geq a_i + d_{i,j} \qquad \forall (v_i, v_j) \in A$$

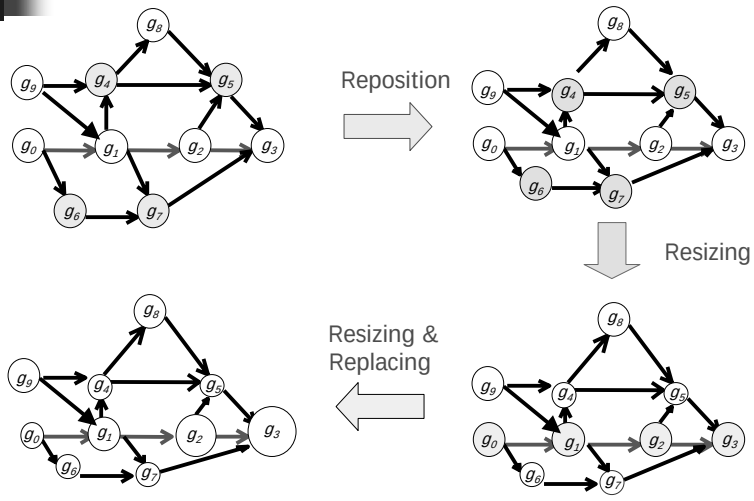
$$a_j \leq \mathrm{T}_{\mathrm{start}} + t_{cycle} \qquad \forall v_j \in PO$$

$$a_j \geq \mathrm{T}_{\mathrm{start}} \qquad \forall v_j \in PI$$

Three Optimizations

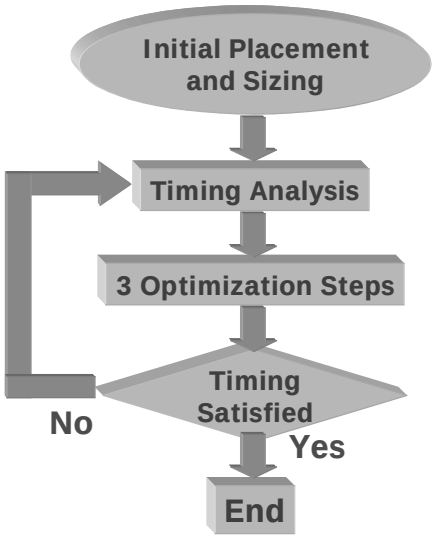
- Reposition the cells directly driven by the cells on the *k* most-critical paths
  - Linear programming (LP)
- Size down the cells directly driven by the cells on the *k* most-critical paths
  - Geometric programming (GP)
- Simultaneously size and place the cells on the *k* most-critical paths
  - Generalized geometric programming (GGP)

Process Steps



Algorithm Flow

- Iteratively select and optimize gates and their immediate fan-outs on the *k* most-critical paths



Review of GGP

- GP formulation - Convex problem

minimize $g_0(x)$

s.t. $g_k(x) \le 0, \quad k = 1, 2, ..., m$

where $g_k(x)$ **is posynomial function, k = 0,1,...,m**

- GGP formulation - Non convex problem

minimize $p_0(x)$

s.t. $p_k(x) \le 0, \quad k = 1, 2, ..., m$

where $p_k(x)$ **is polynomial function, k = 0,1,...,m**

GGP Algorithm

- Transform the original GGP problem into a sequence of (convex) GP problems
- The sequence of optimal solutions to the GP sequence converges to the optimality of the original GGP

Condensation

- Weighted arithmetic-geometric (A-G) mean inequality

$$\sum_i u_i \geq \prod_i \left(\frac{u_i}{\delta_i}\right)^{\delta_i}$$

$$\text{where } u_i > 0,\ \delta_i > 0, \text{ and } \sum_i \delta_i = 1$$

- Polynomial function condensation

$$C[p(x), x'] = \prod_{i=1}^{t} [u_i(x) / \delta_i]^{\delta_i}$$

$$\text{where } p(x) = \sum_{i=1}^{t} u_i(x),\quad \delta_i = \frac{u_i(x')}{p(x')}$$

GGP Transformation

minimize $p_0(x)$

s.t. $p_k(x) \le 0, \quad k = 1, 2, ..., m$

where $p_k(x)$ **is polynomial function,** $k = 0, 1, ..., m$

minimize x_0

s.t. $p_0(x) \le x_0, \; p_k(x) \le 0, \quad k = 1, 2, ..., m$

where $p_k(x)$ **is polynomial function,** $k = 0, 1, ..., m$

minimize x_0

s.t. $g_0^+(x) - g_0^-(x) \le x_0, \; g_k^+(x) - g_k^-(x) \le 0, \quad k = 1, 2, ..., m$

where $g_k^+(x), \; g_k^-(x)$ **is posynomial function,** $k = 0, 1, ..., m$

GGP Transformation, Cont'd

minimize x_0

s.t. $\dfrac{g_0^+(x)}{g_0^-(x) + x_0} \le 1, \; \dfrac{g_k^+(x)}{g_k^-(x)} \le 1, \quad k = 1, 2, ..., m$

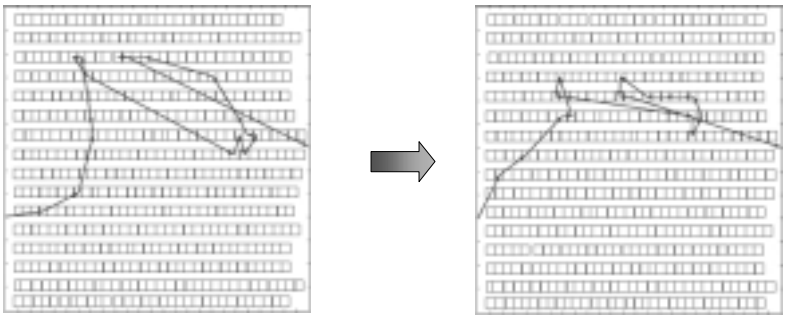
minimize x_0

s.t. $\dfrac{g_0^+(x)}{C[g_0^-(x) + x_0, x']} \le 1, \; \dfrac{g_k^+(x)}{C[g_k^-(x), x']} \le 1, \quad k = 1, 2, ..., m$

- The above is a GP problem
- Original constraints are maintained

Algorithm in Action (I)

- Large freedom of change

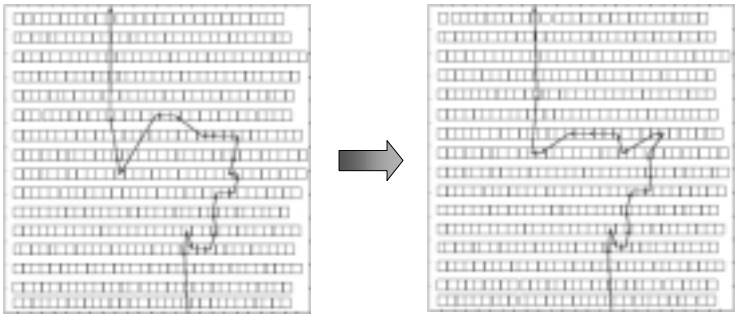


Path delay: 12.43 ns

Path delay: 12.02 ns

Algorithm in Action (II)

- Small freedom of change

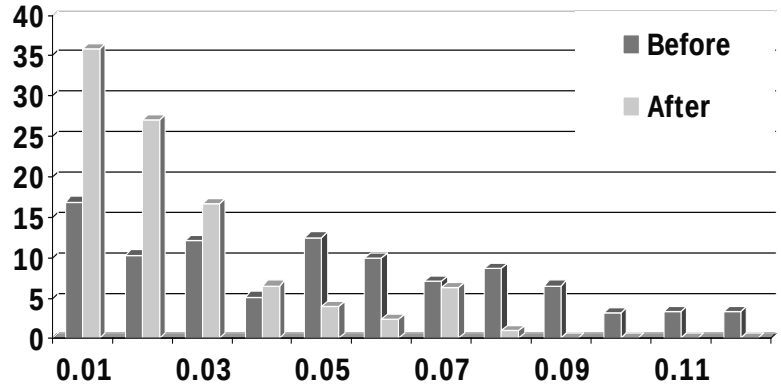


Path delay: 8.27 ns

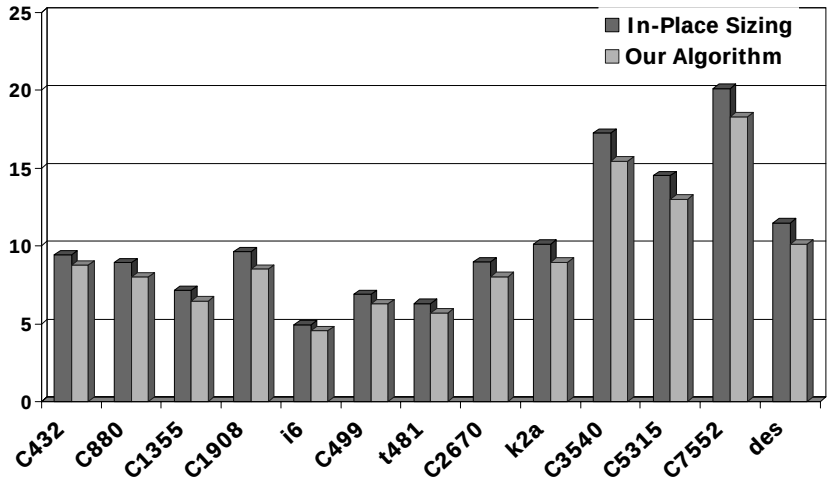
Path delay: 8.22 ns

Comparison of Slack Values

- Normalized slack distribution (C499)
  - X: ratio of the gate slack compared to the longest path delay
  - Y: percentage



Experimental Results



Conclusions

- Our algorithm improves circuit timing by balancing the path delays, i.e., longer delay paths get shorter at the expense of shorter delay paths getting longer
- On average 11% improvement compared to in-place gate sizing

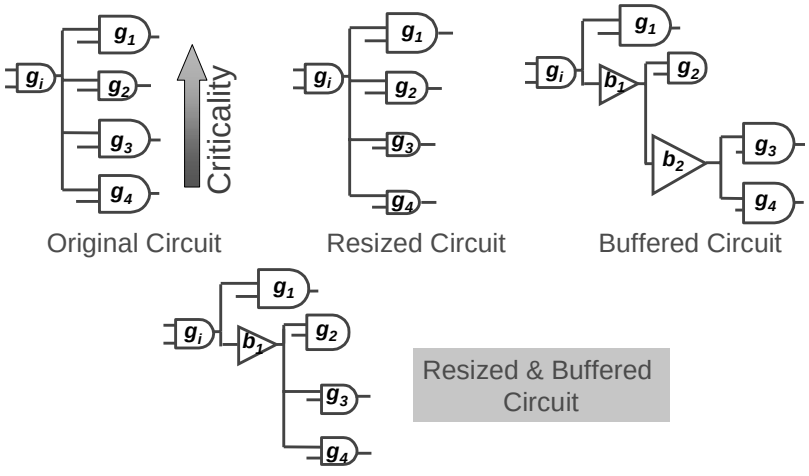
Outline

- Introduction
- Background
- Algorithm
- Other Work
- Future Directions

Other Work

- Concerns
  - Gate sizing maintains the fixed circuit topology
  - Fan-out optimization balances the circuit timing by inserting sized buffers/inverters
- Direction
  - Concurrent gate sizing and fan-out optimization

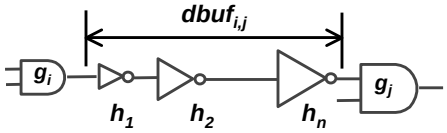
Motivation for Concurrent Gate Sizing and Fan-out Optimization



Motivation Cont'd

- Interleaved Gate Sizing and Fan-out Optimization, Y.Jiang '98
  - For each multi-pin net in the circuit, try out both gate sizing and buffer insertion, and implement the one that yields a better solution
- Integrated Gate Sizing and Fan-out Optimization
  - This is the focus of the work

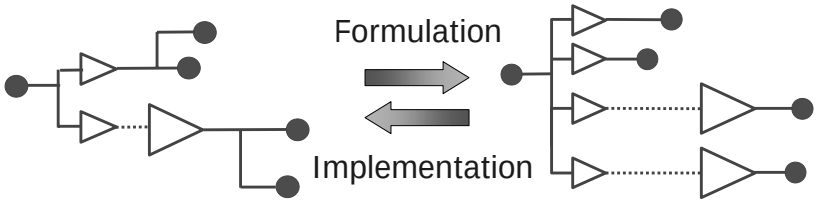
Buffer Insertion Delay Model



- Delay of buffer $d=\tau(p+g\cdot h)$ where p, g and h denote the intrinsic delay, logical effort, and electrical effort, respectively
- Under a required time constraint on g_i, the load of g_i *is* minimized when $h_1=h_2=\ldots=h_n$
- The path delay of the optimal buffer chain is calculated as $\boldsymbol{dbuf_{i,j} = x_{i,j} \cdot (p + g \cdot h_{i,j})}$

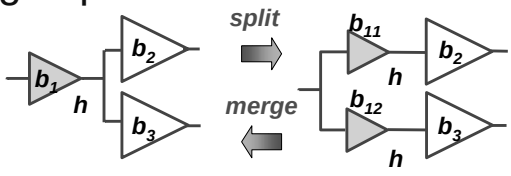
Buffer Tree Formulation

- Difficulty
  - Topology of the buffer tree is unknown
- Solution
  - Recursively split the buffer tree into separate buffer chains

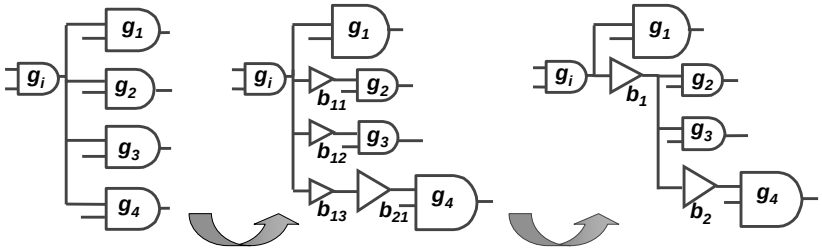


Merge and Split Transformations

- When gains of b_1, b_{11}, b_{12} are the same, the timing and input capacitance properties are preserved by the merge/split transformations



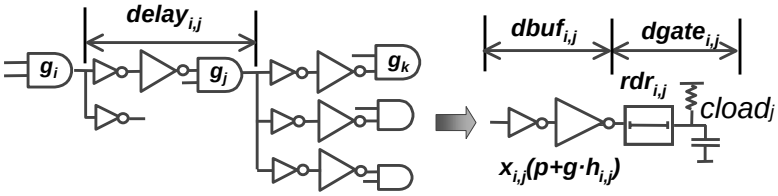
Buffer Tree Construction Example



Size the gates and build the buffer chains

Merge the individual buffer chains

The Complete Delay Model



$$delay_{i,j} = dbuf_{i,j} + dgate_{i,j}$$

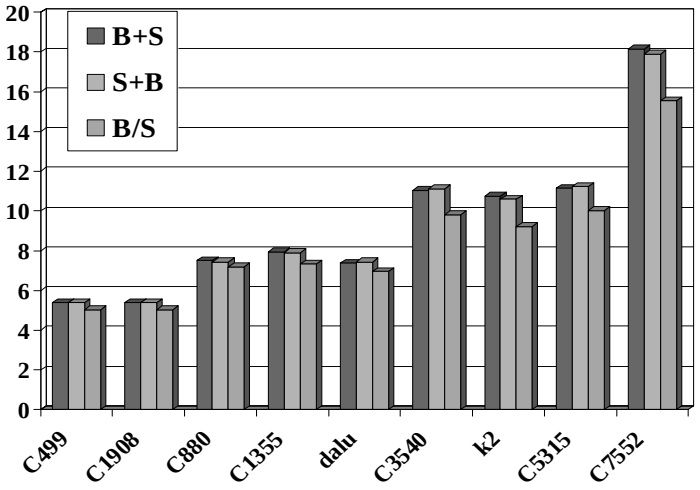
$$dbuf_{i,j} = x_{i,j} \cdot (p + g \cdot h_{i,j})$$

$$dgate_{i,j} = dint_{i,j} + rdr_{i,j}(z_j) \cdot \sum_k \frac{cin_k (z_k)}{(h_{j,k})^{x_{j,k}}}$$

Summary

- Continuous delay model for concurrent gate sizing and buffer insertion
- Iteratively optimize the critical paths
- In each iteration, (1)size the critical gates, (2)build a fan-out tree for the critical gates, (3) size the non-critical fan-out gates of the critical gates simultaneously

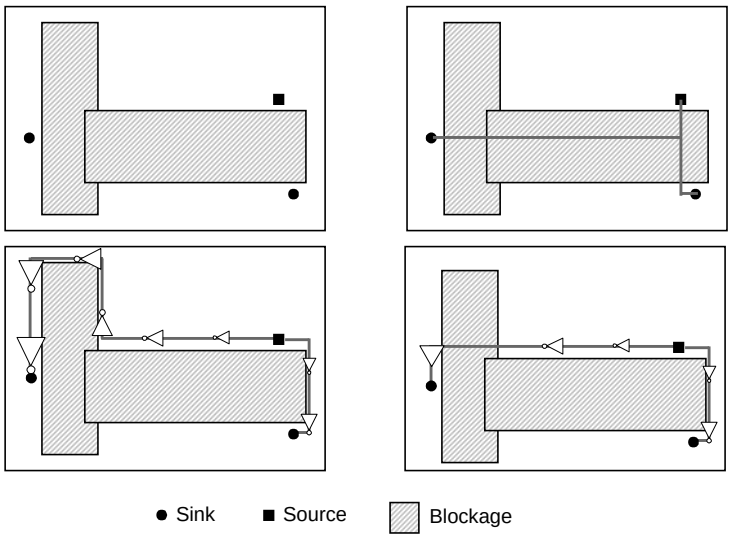
Experimental Results



Outline

- Introduction
- Background
- Algorithm
- Other Work
- Future Directions

Buffered Routing Tree Construction under Buffer Placement Blockages



Problem Definition

- Given (1) a set of placement blockages, where routing is allowed but no buffers can be placed and (2) the locations of the source and the pins of a net, simultaneously build the net topology and insert sized buffers/inverters at the places where they are permitted to improve the timing of the net

Algorithm Outline

- Dynamic programming based
- Generate solutions bottom-up, implement the optimal solution top-down
- Hanan graph
- Line search
- Long edge buffering
- Pruning