

## **PART V**

# **New Approaches to Harness Global Interconnects**

**Jason Cong**

**Computer Science Department  
University of California at Los Angeles**

**Email: [cong@cs.ucla.edu](mailto:cong@cs.ucla.edu)**

**Tel: 310-206-2775**

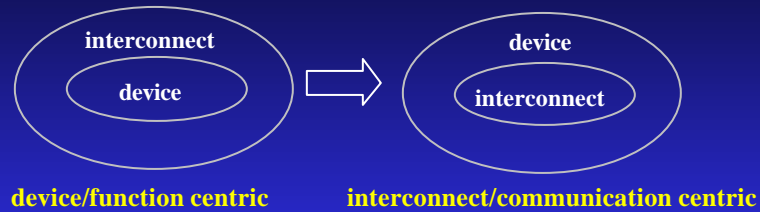
**<http://cadlab.cs.ucla.edu/~cong>**

## **Part V Outline**

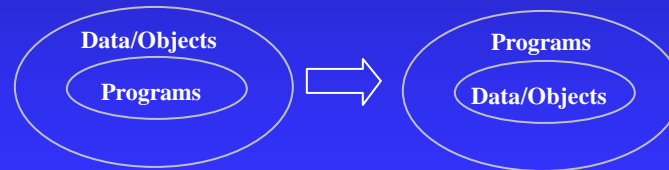
- **Interconnect-Centric Design Flow**
- **Interconnect Performance Estimation**
- **Examples of Interconnect Planning**
  - ◆ **Problem formulation**
  - ◆ **Buffer block planning**
  - ◆ **Wire width planning**
- **System-Level Partitioning with Retiming**
  - ◆ **Hierarchical Performance-Driven Partitioning with retiming**
- **Concluding Remarks**

## Interconnect-Centric Design Methodology

### ■ Proposed transition



### ■ Analogy



## Interconnect-Centric Design Flow

### ■ Key steps in an interconnect-centric design flow:

- ◆ Interconnect Planning
- ◆ Interconnect Synthesis
- ◆ Interconnect Layout

### ■ Other supporting tools to enable an interconnect-centric design flow

- ◆ Interconnect performance estimation
- ◆ Interconnect performance verification

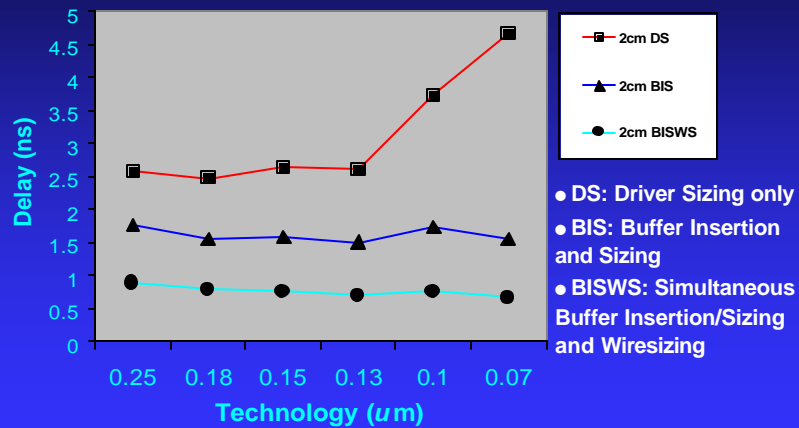
## Interconnect Performance Estimation

- Introduction & Motivation
- Problem Formulation
- Interconnect Delay Estimation Models under Various Layout Optimizations
- Application and Conclusion

## Interconnect Layout Optimization

- E.g., UCLA **TRIO** (Tree, Repeater, Interconnect Optimization) Package
    - ◆ Interconnect topology optimization
    - ◆ Optimal buffer insertion
    - ◆ Wiresizing optimization
    - ◆ Global interconnect sizing and spacing
    - ◆ Simultaneous driver, buffer, and interconnect sizing
    - ◆ Simultaneous topology generation with buffer insertion and wiresizing
- Available from <http://cadlab.cs.ucla.edu/~cong>
- Delay can be improved by up to 7x !

## Impact of Interconnect Optimization on Future Technology Generations



## Complexity of Existing Interconnect Opt. Algorithms

- 2cm line, W=20, B=10, segment every 500um
- Use **best available** algorithms:
  - ◆ Local Refinement (LR)
  - ◆ Dynamic Programming (DP)
  - ◆ Hybrid of DP+LR

Algorithm	LR	DP+LR	DP	
	OWS	BI+OWS	BIWS	BISWS
Delay (ns)	4.5	1.6	1.02	0.81
CPU (s)	0.06	0.42	4.5	12.4

( HSPICE needs additional 60 seconds! )

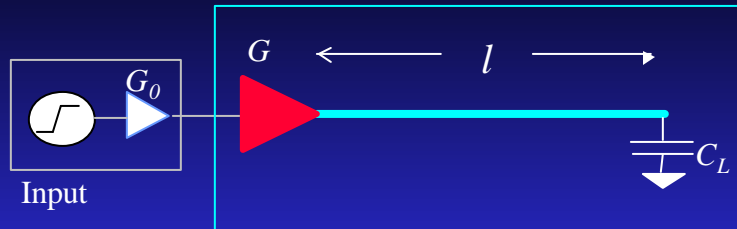
## Needs for Efficient Interconnect Estimation Models

- **Efficiency**
- **Abstraction** to hide detailed design information
  - ◆ granularity of wire segmentation
  - ◆ number of wire widths, buffer sizes, ...
- **Explicit relation** to enable optimal design decision at high levels
- **Ease of interaction** with logic/high level synthesis tools

## Interconnect Performance Estimation Modeling [Cong-Pan, ASPDAC'99, TAU'99, DAC'99]

- Develop a set of **interconnect performance estimation models (IPEM)**, under different optimization alternatives:
  - ◆ Optimal Wire Sizing (**OWS**)
  - ◆ Simultaneous Driver and Wire Sizing (**SDWS**)
  - ◆ Simultaneous Buffer Insertion and Wire Sizing (**BIWS**)
  - ◆ Simultaneous Buffer Insertion/Sizing and Wire Sizing (**BISWS**)
- **IPEM** have
  - ◆ closed-form formula or simple characteristic equations
  - ◆ constant running time in practice
  - ◆ high accuracy (about 90% accuracy on average)

## Problem Formulation



- $R_{d0}$  driver effective resistance of the input stage  $G_0$
- $R_d$  driver effective resistance of  $G$
- $l$  interconnect wire length
- $C_L$  loading capacitance

➔ **What is the optimized delay?**  
**Do not run TRIO or other optimization tools !**

## Parameters and Notations

### ■ Interconnect

- ◆  $c_a$  area capacitance coefficient
- ◆  $c_f$  fringing capacitance coefficient
- ◆  $r$  sheet resistance

### ■ Device

- ◆  $t_g$  intrinsic gate delay
- ◆  $c_g$  input capacitance of the minimum gate
- ◆  $r_g$  output resistance of the minimum gate

- Based on 1997 National Technology Roadmap for Semiconductors (NTRS'97)

## Delay/Area Estimation under OWS

### ■ Closed-form delay estimation formula

$$T_{ows}(R_d, l, C_L) = \left[ \frac{a_1 l}{W^2(a_2 l)} + \frac{2a_1 l}{W(a_2 l)} + R_d c_f + \sqrt{R_d r c_a c_f l} \right] \cdot l$$

where

$$a_1 = \frac{1}{4} r c_a, \quad a_2 = \frac{1}{2} \sqrt{\frac{r c_a}{R_d C_L}}$$

$W(x)$  is Lambert's  $W$  function defined as  $we^w = x$

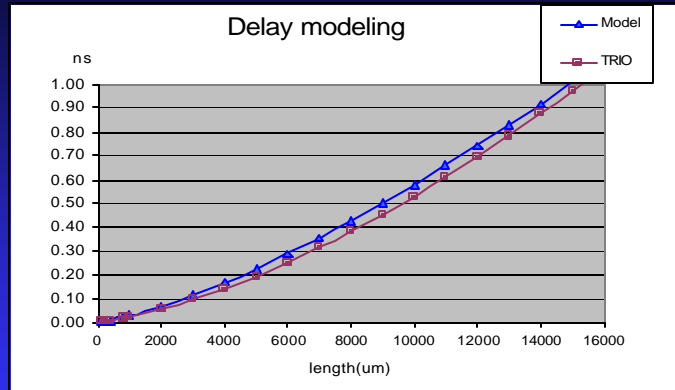
### ■ Closed-form area estimation formula

$$A_{ows}(R_d, l, C_L) = \sqrt{\frac{r(c_f l + 2C_L)}{2R_d c_a}} \cdot l$$

## Property of DEM-OWS

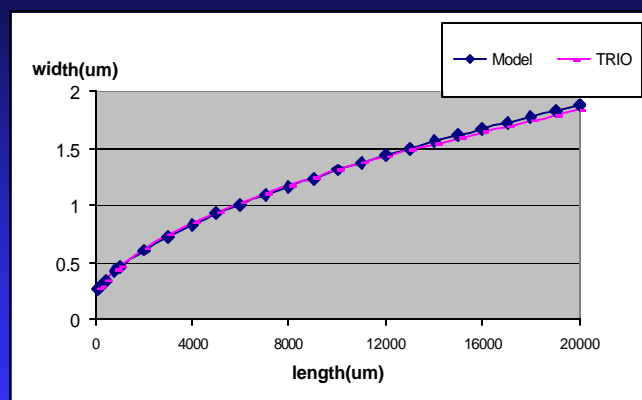
- **Theorem:**  $T_{ows}$  is a sub-quadratic, convex function of length  $l$
- **Note:** Without wiresizing, wiring delay  $\propto l^2$ , as used in some previous layout-driven logic synthesis systems, such as [Ramachandran et al., ICCAD-92] – no longer accurate!
- Closed-form DEM-OWS will serve as a basis for deriving SDWS, BIWS and BISWS

## Comparison of IPEM-OWS vs. TRIO



- $0.18\mu\text{m}$ ,  $R_d = r_g/100$ ,  $C_L = c_g \times 100$
- For expt., max wire width is 20x min, wire is segmented in every 10um

## Area Estimation for OWS

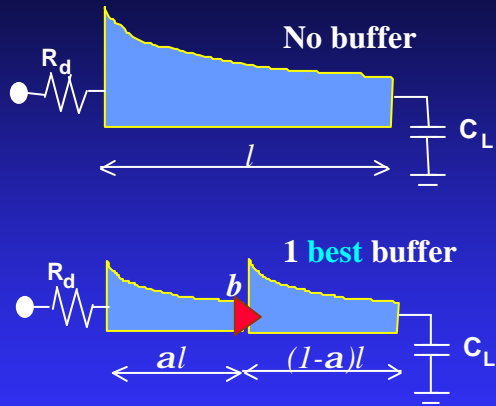


## Critical Length for BI under OWS

$$T_{ows}(R_d, l, C_L)$$

Solve for  $l$ , =>  
critical length  $l_{crit}$   
( $b, R_d, C_L$ )

- Computed by bisection method
- Constant time in practice



$$T_{biws}(R_d, l, C_L) = \min_{0 \leq a \leq 1} \{T_{ows}(R_d, a l, C_b) + t_g + T_{ows}(R_b, (1-a)l, C_L)\}$$

## Critical Lengths $l_{crit}(b, R_b, C_b)$

Decrease

Technology (um)	0.25	0.18	0.15	0.13	0.10	0.07
b=10x	4.12	3.80	3.97	3.61	2.92	2.08
b=50x	6.40	5.81	6.01	5.51	4.45	3.30
b=100x	7.47	6.83	7.04	6.39	5.30	3.91
b=200x	8.65	7.92	8.14	7.43	6.35	4.49
b=500x	9.98	9.10	9.30	8.57	7.13	5.21

unit: mm

Min. WS	2.52	2.23	2.14	1.94	1.50	1.43
---------	------	------	------	------	------	------

- Cf. [Otten ISPD'98, Otten-Brayton DAC'98]  
(uniform wire width)

- Denote  $l_c = l_{crit}(b, R_b, C_b)$

## “Logic Volume” within $l_c$

- Defined as the number of min 2-input NAND gates that can be packed within the area of  $l_c/2 * l_c/2$

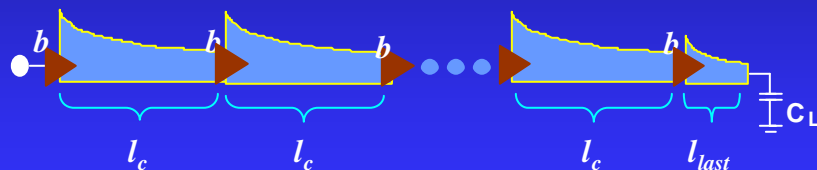
Technology (um)	0.25	0.18	0.15	0.13	0.10	0.07
2-NAND (um <sup>2</sup> )	7.80	4.04	3.00	2.18	1.28	0.64
b=10x	0.55	0.89	1.31	1.49	1.66	1.69
b=50x	1.31	2.09	3.01	3.48	3.87	4.25
b=100x	1.79	2.88	4.13	4.68	5.48	5.97
b=200x	2.4	3.88	5.52	6.33	7.87	7.88
b=500x	3.19	5.12	7.21	8.42	9.93	10.6

unit: million

→ Increase

## Property of BIWS

- **Theorem:** For BIWS, the distances between adjacent buffers are the same, and equal to  $l_c$  -- the critical length.
- **Proof:** based on the convexity of  $T_{ows}$



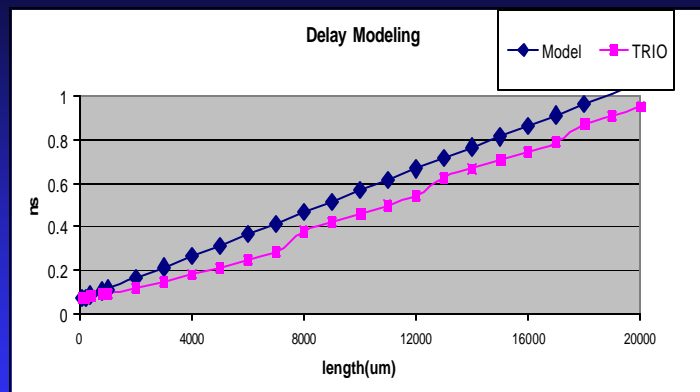
## IPEM for BIWS

- Original long interconnect is divided into  $l/l_c$  stage
- The **stage number** is proportional to  $l$
- Each stage of length  $l_c$  has delay  $T_{ows}(R_b, l_c, C_b)$
- ➔ Linear DEM for BIWS

$$T_{biws} = t_{biws} \cdot l + t_g$$

$t_{biws}$  is the slope, and can be obtained from  $T_{ows}(R_b, l_c, C_b)$

## IPEM for BIWS vs. TRIO



- $0.18\mu m$ ,  $R_{d0} = r_g/10$ ,  $C_L = c_g \times 10$ , buffer type is 100 x min.
- For expt., max. wire width is 20x min. width, wire is segmented in every 100um.

## IPEM under BISWS

- Observations from **extensive** experiments:
  - ◆ Linear delay versus length
  - ◆ Internal buffers are about the same size
- Therefore, we estimate BISWS by the best BIWS from available buffer types

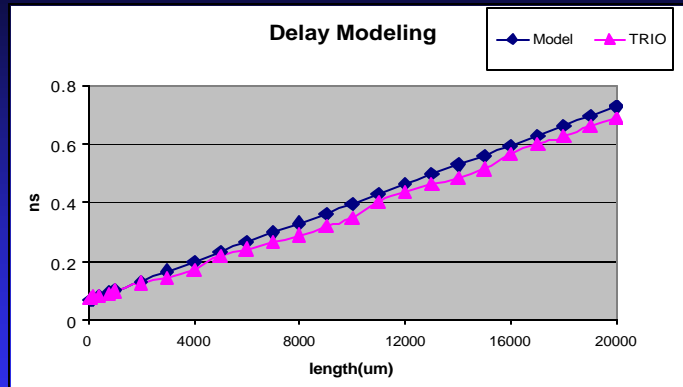
- Linear delay model for optimal BISWS

$$T_{bisws} = t_{bisws} \cdot l + t_g$$

where  $t_{bisws} = \min_{b \in B} t_{biws}$ ,  $B$  is the buffer set

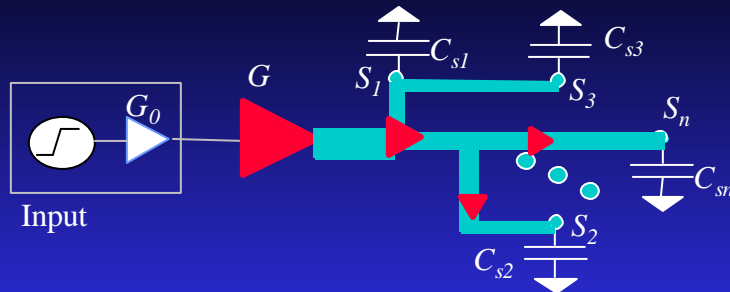
- Complexity  $O(|B|)$ . Since the set  $B$  is normally less than 20, constant time in practice.

## Comparison of IPEM for BISWS vs. TRIO



- $0.18\mu m$ ,  $R_{d0} = r_g / 10$ ,  $C_L = c_g \times 10$
- For expt., max. allowable buffer/driver size is 400x min device; max. wire width is 20x min. width; wire is segmented in every 100um.

## IPEM for Multiple-Pin Nets



- Estimation with different optimization objectives:
  - ◆ Minimize the delay to a single critical sink (SCS)
  - ◆ Minimize the maximum delay (defined as the tree delay) for multiple critical sinks (MCS)
  - ◆ Minimize weighted delay ...

## Challenges for Multiple-Pin Net Estimation

- No closed-form wire shaping function available
- Current optimization algorithms
  - ◆ Iterative based method
    - ◆ Local refinement
    - ◆ Dynamic Programming
    - ◆ Lagrangian relaxation
    - ◆ Mathematical programming
  - ◆ Not suitable for estimation



**Key idea: transform to 2-pin net !**

## Basic Approach

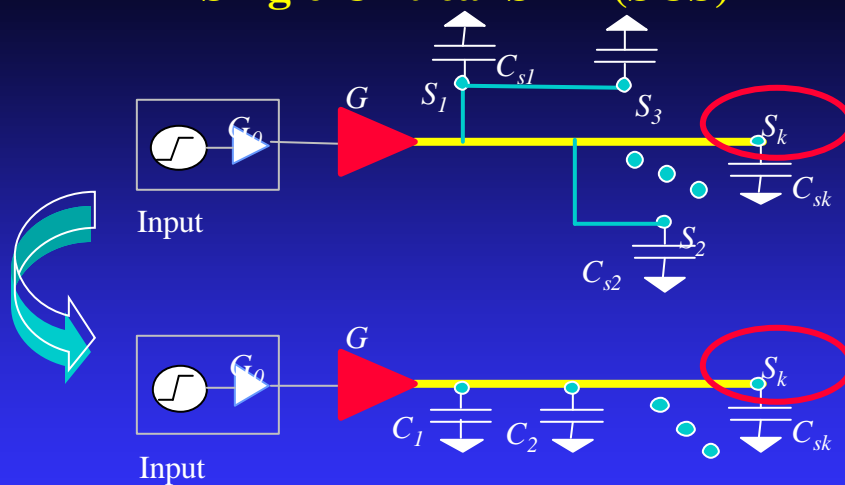
### ■ Estimation for Single Critical Sink

- ◆ We first formulate the original problem into a single-line-multiple-load (SLML) problem
- ◆ Then transform SLML into a single-line-single-load (SLSL) problem
- ◆ Use previous 2-pin results to estimate delay and area on the critical path

### ■ Estimation for Multiple Critical Sinks

- ◆ We obtain a lower bound delay estimation for the optimal tree delay
- ◆ We show that in practice, the above lower bound estimation is tight and close to the optimal tree delay

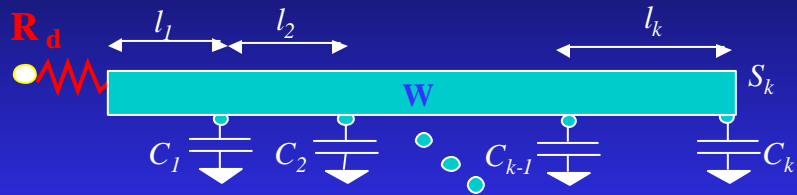
## Single Critical Sink (SCS)



## Single-Line-Multiple-Load

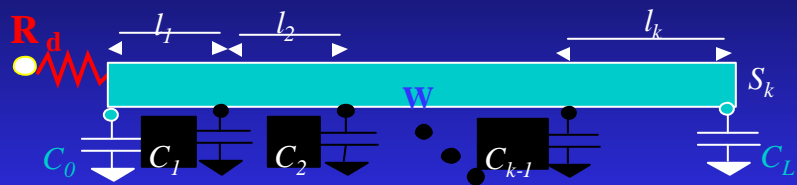
## OWS for SCS

- Transform SLML to SLSL (i.e., 2-pin net)



## OWS for SCS

- Transform SLML to SLSL (i.e., 2-pin net)



$$C_L = \sum_{j=1}^k \frac{\sum_{i=1}^j l_i}{l} \cdot C_j \quad C_0 = \sum_{j=1}^k C_j - C_L$$

## Reduced to 2-Pin Problems

- Closed-form delay estimation for the critical sink

$$T_{ows}(R_d, l, C_L) = R_d C_0 + \left[ \frac{a_1 l}{W^2(a_2 l)} + \frac{2a_1 l}{W(a_2 l)} + R_d c_f + \sqrt{R_d r c_a c_f l} \right] \cdot l$$

where

$$a_1 = \frac{1}{4} r c_a, \quad a_2 = \frac{1}{2} \sqrt{\frac{r c_a}{R_d C_L}}$$

$W(x)$  is Lambert's  $W$  function defined as  $we^w = x$

- Closed-form area estimation for the critical path

$$A_{ows}(R_d, l, C_L) = \sqrt{\frac{r(c_f l + 2C_L)}{2R_d c_a}} \cdot l$$

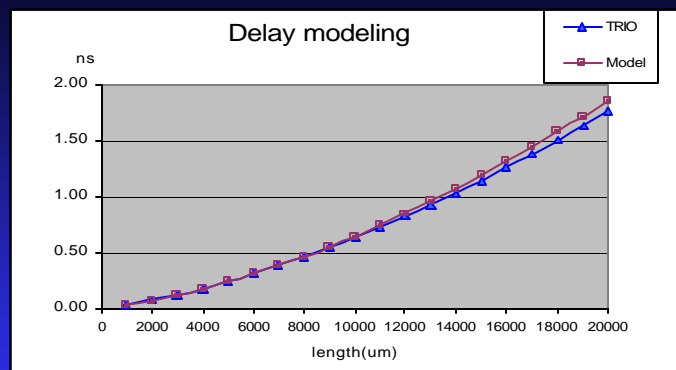
## Multiple Critical Sinks (MCS)

- Optimization objective: the maximum delay to all critical sinks, i.e. the tree delay
- **Key idea:** transform MCS to a sequence of SCS
- **Theorem:** The most critical sink with max delay must be a leaf critical sink.
- **Theorem:** The optimal delay to any critical sink under SCS formulation is a lower bound for the optimal tree delay.

## Multiple Critical Sinks/OWS

- **Key observation:** take the **maximum delay** of all **leaf critical sinks** under SCS formulation => accurately estimate the optimal tree delay
- **Justification:** we shall keep wire load from less critical sinks as small as possible. To the most critical sink, the main difference is
  - ◆ (A) 'minimum width' under SCS formulation
  - ◆ (B) 'as small as possible width' under MCS formulation
  - ◆ In DSM, area capacitance is relatively small (cf. fringing + coupling cap.) => Two wire loads (A) and (B) differ not much.

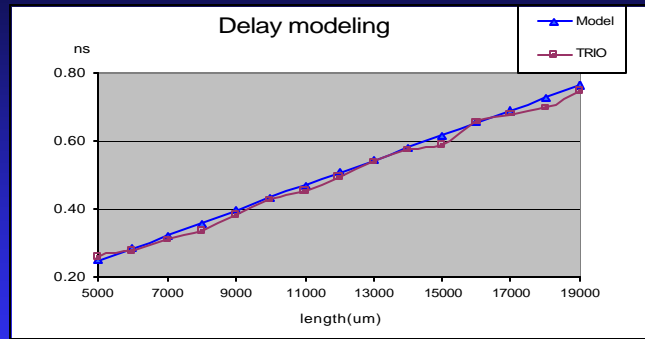
## Multiple Critical Sinks/OWS



- Random 4-pin nets,  $0.18\mu\text{m}$  tech,  $R_d = 180\text{ohm}$ ,  $C_s = 10\text{fF}$
- TRIO uses max. allowable wire width of  $20\times$  min; wire is segmented in every  $500\mu\text{m}$ .
- Length is the distance from source to 'most critical' sink

## MCS/BISWS

- Similar to OWS, take the max of SCS/BISWS



- Random 4-pin nets ,  $0.18\mu m$ ,  $R_{d0} = r_g/10$ ,  $C_s = c_g \times 10$
- TRIO uses max. buffer size of 400x min, wire width of 20x min. width; wire is segmented in every 500um.

## Some Applications of IPeM

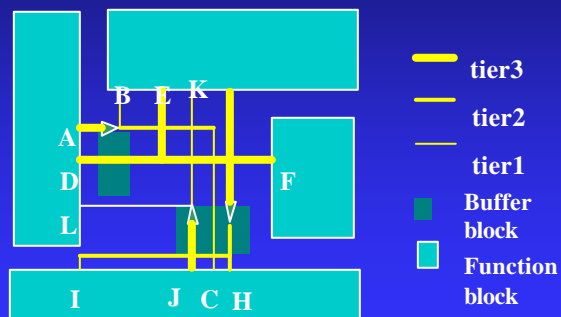
- Layout-driven physical and RTL level floorplanning
  - ◆ Predict accurate interconnect delay and routing resource without really going into layout details;
  - ◆ Use accurate interconnect delay/area to guide floorplanning/placement
- Interconnect Architecture Planning
  - ◆ E.g. Wire width planning
- Floorplanning + interconnect planning
  - ◆ E.g. Buffer block planning
- Available from <http://cadlab.cs.ucla.edu/~cong>

## Part V Outline

- Interconnect-Centric Design Flow
- Interconnect Performance Estimation
- Examples of Interconnect Planning
  - ◆ Problem formulation
  - ◆ Buffer block planning
  - ◆ Wire width planning
- System-Level Partitioning with Retiming
  - ◆ Hierarchical Performance-Driven Partitioning with retiming
- Concluding Remarks

## Interconnect-Driven Floorplanning

- Planning location and dimensions of each major function block
- Plan for layer, topology, width, and spacing for critical interconnects

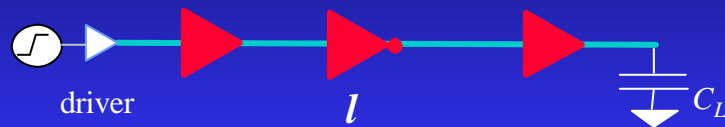


## Buffer Block Planning in Interconnect-Driven Floorplanning

- Buffer Block Planning Algorithm [Cong-Kong-Pan, ICCAD'99]
- Routability-Driven Repeater Block Planning for Interconnect-Centric Floorplanning [Sarkar-Sundaraman-Koh, ISPD'00]
- Planning Buffer Location by Network Flows [Tang-Wong, ISPD'00]

## Buffer Insertion

- Interconnect dominates transistor delay for deep sub-micron designs.
- Buffer insertion is a very effective way to trade active devices for better interconnect performance, noise reduction, etc.



- Without buffer: delay  $\mu l^2$  or  $\mu l^2/W(l)$  [Cong-Pan'98]
- With opt. Buffers: delay  $\mu l$  [Bakoglu'90; Otten-Brayton'98; Cong-Pan'98]

## Demand of Buffers in DSM Design

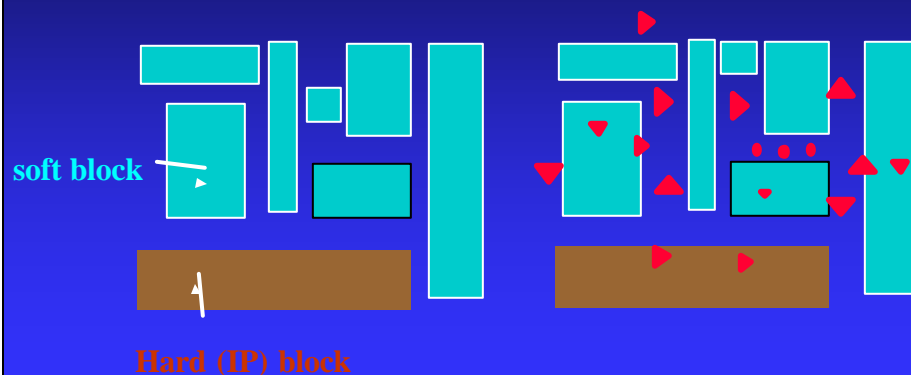
- For high-performance DSM designs, many buffers may be inserted to optimize/meet interconnect delay

<i>Technology (um)</i>	<i>0.25</i>	<i>0.18</i>	<i>0.13</i>	<i>0.10</i>	<i>0.07</i>
<b>#buffer per chip</b>	5k	25k	54k	230k	797k

Source: [Cong'97, SRC Work Paper]  
<http://www.src.org/research/frontier.dgw>  
( Data based on NTRS'97)

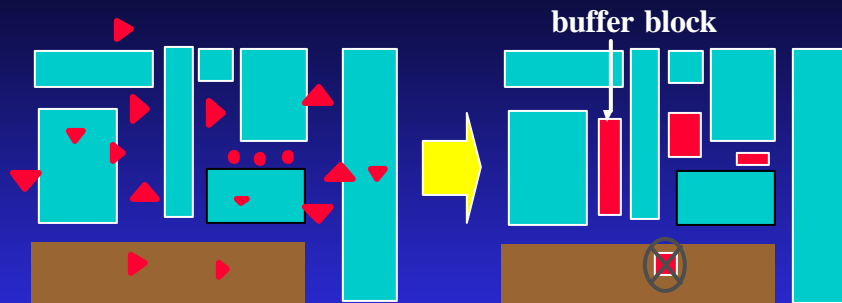
## Need Buffer Planning

- The insertion of so many buffers will significantly change a floorplan; thus shall be planned ahead-of-time to ensure timing/design convergence.



# Buffer Block Planning

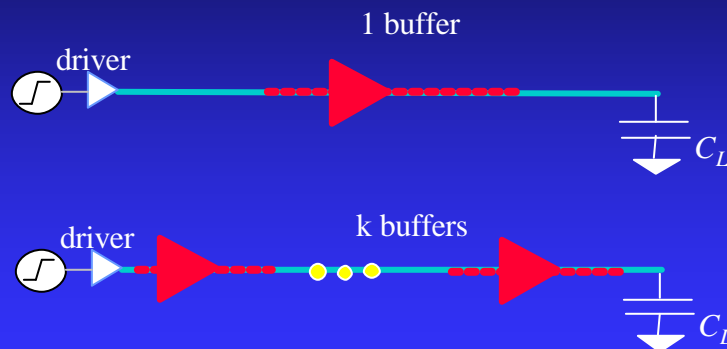
[Cong-Kong-Pan, ICCAD'99]



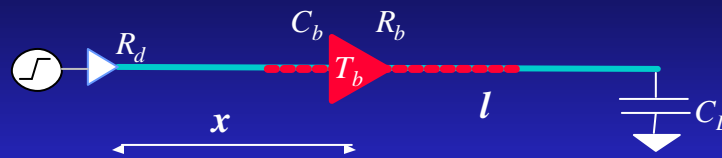
- **Given:** initial floorplan and performance constraint for each net
- **Output:** “optimal” location/dimension of buffer blocks such that the overall chip area and the number of buffer blocks are minimized

# Feasible Region for BI

- **Feasible region** is the maximal region that a buffer can be placed to meet given delay constraint.



## Feasible Region for One Buffer

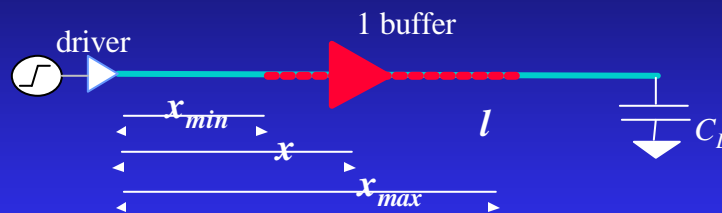


- Feasible region condition for  $x$

$$T_0(R_d, x, C_b) + T_b + T_0(R_b, l - x, C_L) \leq T_{req}$$

## Feasible Region for One Buffer

- We obtain **closed-form** formula of FR for inserting one buffer to meet delay constraint



$$x \in [x_{min}, x_{max}]$$

## Feasible Region for One Buffer

$$T = Ax^2 + Bx + C$$

where  $A = rc$

$$B = (R_d - R_b)c + r(C_b - C_L) - rcl$$

$$C = R_d C_b + R_b C_L + (R_b c + r C_L)l + \frac{1}{2} r c l^2 + T_b$$

Optimal buffer location:  $x^* = -B / 2A$

Let:  $x = x^* + d$

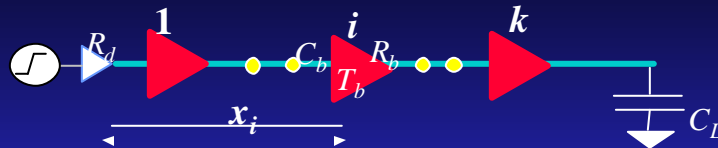
$$\begin{aligned} T &= A(x^* + d)^2 + B(x^* + d) + C \\ &= Ax^{*2} + Bx^* + d + Ad^2 + 2Ax^*d + Bd \\ &= T_{opt} + Ad^2 \leq T_{req} \end{aligned}$$

$$\Rightarrow d \leq \sqrt{\frac{T_{req} - T_{opt}}{rc}}$$

$$x_{min} = x^* - d$$

$$x_{max} = x^* + d$$

## FR for Multiple Buffers

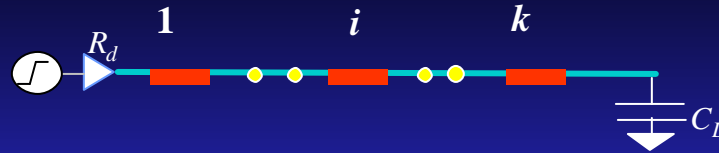


- Feasible region condition for the  $i$ -th buffer  $x_i$

$$T_{i-1}(R_d, x_i, C_b) + T_b + T_{k-i}(R_b, l - x_i, C_L) \leq T_{req}$$

- More complicated, but still **closed-form solution** for FR can be obtained.
- We also obtain the minimum number of buffers  $k_{min}$  needed to meet delay constraint

## Independent FR for Multiple Buffers



- **Independent Feasible Region (IFR)** [Sarkar-Sundararaman-Koh, ISPD'00] is a **subset** of FR [Cong-Kong-Pan, ICCAD'99]
- Independent of each other in **one-dimensional** nets, but need to maintain monotonic path for two-dimensional nets (thus not totally independent).

■ IFR width  $d \leq \sqrt{\frac{T_{req} - T_{opt}}{(2n - 1)rc}}$

$x_{i \min} = x_i^* - d$   
 $x_{i \max} = x_i^* + d$

## Different Feasible Regions

- **FR**: [Cong-Kong-Pan, ICCAD'99] the **maximal** region that a buffer can be placed to meet given delay constraint (assume all other buffers can be **optimally** re-placed)
- **Restricted FR (RFR)**: a subset of FR that assumes all other buffers are **fixed** in their original delay-minimal positions.

$$d_{RFR} = \sqrt{\frac{T_{req} - T_{opt}}{rc}}$$

- **Independent Feasible Region (IFR)** [Sarkar-Sundararaman-Koh, ISPD'00]: a **subset** of both FR and RFR

$$d_{IFR} \leq \sqrt{\frac{T_{req} - T_{opt}}{(2n - 1)rc}}$$

## FR versus IFR

- For 1 buffer, the same
- For multiple buffers, IFR is just a small fraction of FR
  - ◆ Usually more buffers inserted for a net => smaller IFR/FR
  - ◆ For example of 4 buffers inserted, can be IFR/FR can be  $\frac{1}{4}$

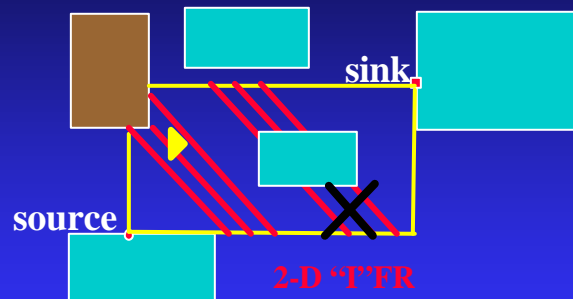
Critical length is 4269um for BI

length (um)	budget	#buffer	FR (um) of each buffer	RFR (um)	IFR (um)	IFR/FR for each buffer
7000	5%	1	2659	2659	2659	1
7000	10%	1	3760	3760	3760	1
14000	5%	3	4501; 5197; 4501	3675	1643	0.365; 0.316; 0.365
14000	10%	2	4312; 4312	3734	2155	0.5; 0.5
20000	5%	4	4441; 5439; 5439; 4441	3511	1327	0.299; 0.244; 0.244; 0.299
20000	10%	3	4268; 4928; 4268	3484	1558	0.365; 0.316; 0.365

1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup>, 4<sup>th</sup> buffers

## IFR Not Independent in 2D

- FR extended to 2-dimension with obstacles

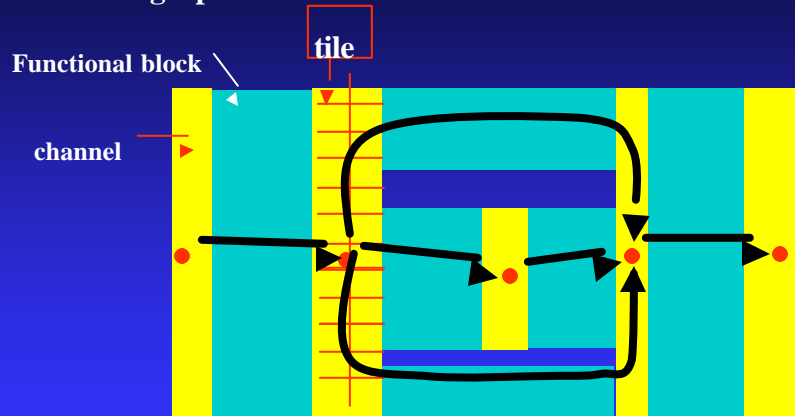


## BBP for Interconnect-Driven Floorplanning

- For each floorplan (*FL*) configuration
  - ◆ Apply BBP on the given *FL*
  - ◆ Evaluate resulting *FL* in terms of timing, area, #BB trade-off, etc.
- Return the best *FL* solution

## Data Model for BBP in FL

- Polar graph and tile structure



- Both slicing and non-slicing floorplans

## Overview of BBP Algorithm

1. Build polar graph/tile for given floorplan  $FL$ ;
2. For each tile, compute its area slack;
3. Compute FR for each buffer of each net;
4. While (there exists some buffer to be inserted) {  
    Pick\_A\_Tile  $t$ ;  
    Insert\_Buffers into  $t$ ;  
    Update chip dimension, FR, area slack, etc.  
}

## Overview of RBP Algorithm

1. Build tile structure for given floorplan  $FL$ ;
2. Compute IFR for each repeater and obtain candidate set for each repeater;
3. Generate bipartite graph  $G = (b,c)$ ;
4. While (there exists some buffer to be inserted) {  
    Delete highest cost edge of  $G$ ;  
    Update monotonicity, congestion matrix, and edge costs;  
    Assign repeater to CRB if required;  
}

## Bipartite Graph G

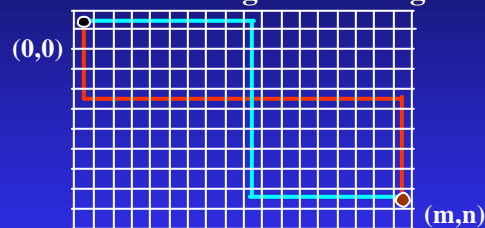
$G = (b, c)$ , where **b** is a buffer and **c** is a candidate buffer block from IFR

The **edge weight**  $(b,c)$  reflects the “cost” of assigning the repeater **b** to the buffer block **c**

**Cost function:** composition of routing congestion and #BB

## Congestion Model

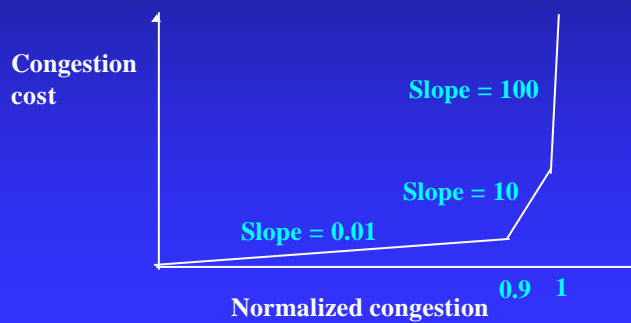
- Similar to [Chen+, ICCAD'99] which is essentially a two dimensional rectangular grid based probabilistic map
- Assume **two bend** routing for each segment



- Total two bend routes:  $(m+n+2)$
- Then the probability of a horizontal/vertical route pass through tile  $(i,j)$  can be computed

## Congestion Model

- Compute the horizontal/vertical congestion matrices by summing up all nets and buffers.
- Then the congestion cost is assigned based on the congestion normalized matrices in a PWL manner [Cong-Madden, DAC'98]



## Dynamic Edge Weights

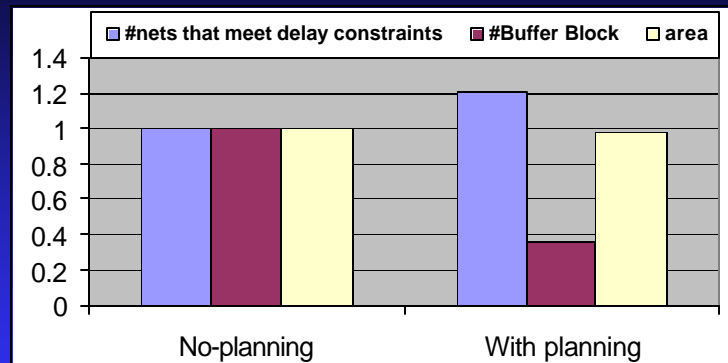
- Edge weights for bipartite assignment graph consist of both congestion cost (CC) and repeater block cost (BB)
- Congestion Cost (CC) is the maximum congestion cost among all routing tiles in the one-bend routing path from source (or previous repeater) to sink (or next repeater)
- Repeater Block Cost (BB):  $1/\min(N_c, N_{\max})$ , where  $N_c$  is the #IFRs intersecting with this tile,  $N_{\max}$  is the capacity. Note: channel expansion not allowed, i.e.,  $BB(c) = \infty$  if exceed capacity.

$$COST(e) = (CC(e))^{p1} \cdot (BB(c))^{p2}$$

## Iterative Deletion

- 1. Iteratively delete a **single** high-cost assignment at each step, then update
- 2. Monotonicity updates (make sure monotone path)
- 3. Congestion updates after deleting the highest cost edge (because it precludes some b->c option)
- 4. Edge cost updates due to the monotonicity and congestion updates and go to step 1

## Experimental Results of Buffer Block Planning [Cong, et al, ICCAD'99]



Buffer block planning reduces # buffer blocks, better meets timing constraints, and use smaller area

## Planning Buffer Location by Network Flows [Tang-Wong, ISPD'00]

- Assume given floorplan and dead area, insert maximum number of buffers into the free space
- Objective: insert the maximum number of buffers into free space
- Solution: min-cost network-flow formulation
- Limitations:
  - ◆ How to allocate 'free space'
  - ◆ Only one buffer per net is considered

## Wire Width Planning (WWP) [Cong-Pan, DAC'99]

- Given:
  - ◆ Certain technology
  - ◆ Wire assignment for each metal layer
- Find:
  - ◆ A small set of "globally optimal" widths per layer
  - ◆ Performance/Area optimization
- Motivation
  - ◆ Simplify interconnect optimization
  - ◆ Ease routing architecture

## Design Optimization Objective

- Given some weight function  $I(l)$  for wire length range  $[l_{min}, l_{max}]$ , we want to find *a small set of optimal widths*  $\vec{W}^*$  to minimize

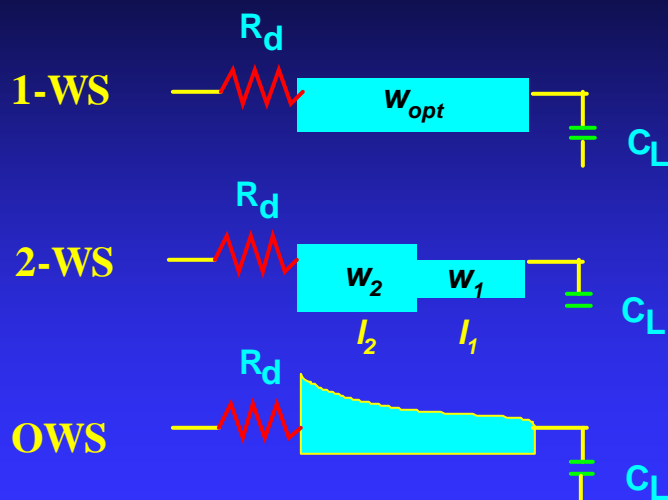
$$\Phi(\vec{W}, l_{min}, l_{max}) = \int_{l_{min}}^{l_{max}} I(l) \cdot f(\vec{W}, l) dl$$

- $f(\vec{W}, l)$ : the objective function to be minimized by the design using  $\vec{W}$

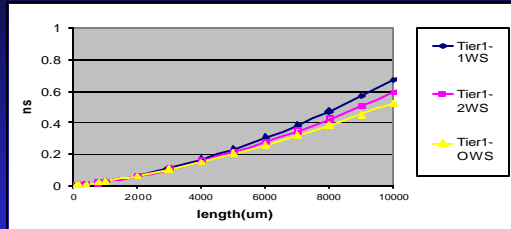
$$f(\vec{W}, l) = A^j(\vec{W}, l) \cdot T^k(\vec{W}, l) \quad (A\text{-area } T\text{-delay})$$

- or
- $f(\vec{W}, l) = T(\vec{W}, l)$  (performance only)
  - $f(\vec{W}, l) = A(\vec{W}, l) \cdot T^4(\vec{W}, l)$  (performance-driven and area-saving)

## Two Simplified Wire-sizing Schemes



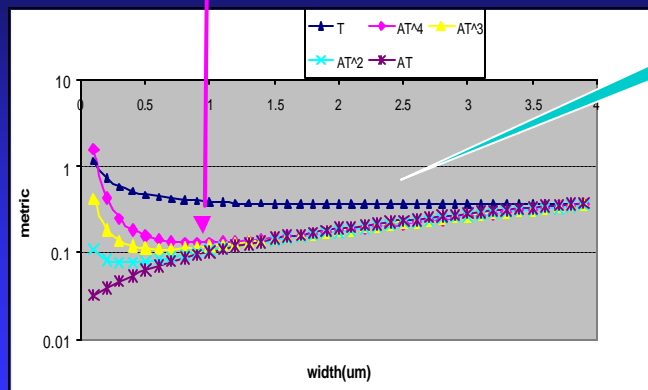
## 1-WS and 2-WS



- 1-WS and 2-WS have less than 10% difference from OWS for length <4mm in Tier1 (Metal 1-2)
- Both work well in upper metal layer up to chip dimension
- In above figure, **different widths** for different lengths.

## A Performance-Driven, Area-Saving Metric

Opt. width for  $AT^4$ . Only increase delay by 10%, save area by 60%!



Optimal width for delay T

- 0.10um tech;
- Top layer pair;
- Length range 8 -23 mm;
- Assume equal weight;
- Metric: integral of T, AT, AT<sup>2</sup>, ..., AT<sup>4</sup>
- Driver/load 100x min gate

## Overall Approach

For each metal layer  $i$

Assign length range  $l_{min}$  and  $l_{max}$ ;  
Find  $W^*$  or  $\{W_1^*, W_2^*\}$  to minimize

$$\Phi(\bar{W}, l_{min}, l_{max}) = \int_{l_{min}}^{l_{max}} I(l) \cdot f(\bar{W}, l) dl$$

**Method: Analytical and numerical**

## Overall Approach (Cont'd)

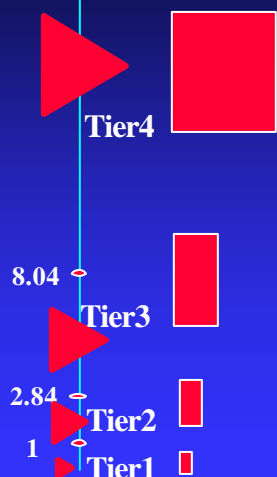
- Analytical formula for 1-width planning for best T

$$W^* = \sqrt{\frac{\frac{1}{3} rC_f (l_{max}^3 - l_{min}^3) + rC_L (l_{max}^2 - l_{min}^2)}{RdCa (l_{max}^2 - l_{min}^2)}}$$

- Numerical method by effective searching the best 1-width planning under AT<sup>4</sup> metric and the best 2-width planning under T and AT<sup>4</sup> metrics

## Experimental Setting

22.8 mm  
(0.10um tech.)



- Parameters based on NTRS'97 and Strawman [Otten&Brayton, DAC'98]
- Assume uniform weight function and the max length in Tier1 is 10,000x feature size, and in top tier is the chip dimension
- Intermediate tier's length range follows a geometric sequence
- Representative driver size for each metal layer (10x, 40x, 100x, and 250x for tiers 1-4)
- Verify against optimal wire sizing and spacing algorithm (using many widths) [Cong+, ICCAD'97]

## Surprising Result: Two widths good enough! [Cong-Pan, DAC'99]

scheme	pitch-sp=2um			pitch-sp=2.9um			pitch-sp=3.8um		
	avg-d	max-err	avg-w	avg-d	max-err	avg-w	avg-d	max-err	avg-w
1-width	0.245	28%	1.98	0.177	16%	1.83	0.143	6%	1.63
2-width	0.215	7%	1.08	0.167	5.90%	1.23	0.14	4%	1.41
m-width	0.204	0%	1.03	0.159	0%	1.19	0.136	0%	1.38

- Case study of 0.10um using upper metal pair
- 2-width design superior than 1-width design
  - delay reduction up to 12.4%
  - area saving up to 48% !
- 2-width design comparable to many-width design
  - Avg. delay less than 5% and Max. delay less than 7%
  - Area difference less than 4.7%

## Max-error Theorem

$$\Phi(\vec{W}, l_{\min}, l_{\max}) = \int_{l_{\min}}^{l_{\max}} I(l) \cdot f(\vec{W}, l) dl$$

If  $\frac{f(\vec{W}, l) - f(\vec{W}^*, l)}{f(\vec{W}^*, l)} \leq d_{\max}$  for any  $l \in (l_{\min}, l_{\max})$

⇒  $\left| \frac{\Phi(\vec{W}, l_{\min}, l_{\max}) - \Phi(\vec{W}^*, l_{\min}, l_{\max})}{\Phi(\vec{W}^*, l_{\min}, l_{\max})} \right| \leq d_{\max}$  for any  $I(l)$

- Max-error of any length is less than 7% in our expt =>
- For **any** weight function  $I(l)$ , the max-error of weighted integral (our objective function) is less than 7% !

## Sample WWP for Future Tech.

[Cong-Pan, DAC'99]

- 2-width design under objective function of  $AT^4$
- Wiring hierarchy for both performance and density !

Technology (um)		0.25	0.18	0.13	0.10	0.07
M12	W (um)	0.25	0.18	0.13	0.10	0.07
M34	W1(um)	0.25	0.18	0.13	0.10	0.08
	W2(um)	0.50	0.36	0.26	0.20	0.16
M56	W1(um)	0.65	0.47	0.24	0.22	0.23
	W2(um)	1.30	0.94	0.48	0.44	0.46
M78	W1(um)	-	-	0.98	1.00	1.06
	W2(um)	-	-	1.96	2.00	2.12

## Estimating/Optimizing Routing Utilization [Chong-Brayton, SLIP'99]

- Purpose: system level estimation/optimization for routing utilization
- Use stochastic wire length model, based on the **Rent's rule** and **interconnect density function** from [Davis-De-Meindl, IEEE TED'98]
- Greedy layer assignment with delay constraint:
  - ◆ Tier-based (one-tier) routing for each net
  - ◆ Shorter wires assigned to lower tiers and longer wires to higher tiers, in a greedy manner
  - ◆ The longest wire for each tier is computed based on the delay constraint, with consideration of buffer insertion [Otten-Brayton, DAC'98]

## Estimating/Optimizing Routing Utilization Results [Chong-Brayton, SLIP'99]

- 0.10um process, 80ps delay constraint
- Longest wire in gate pitches; metal widths in 1 units
- Best wire widths for each tier are obtained (by enumeration) to optimize the routing utilization at each tier (while meeting delay target)

Design gates	Longest wire	Metal widths	longest metal	unrouted
4M	3457	2/3/8/20/23	3457	0
5M	3883	2/3/7/16/26	3883	0
6M	4270	2/3/7/14/28	4270	0
7M	4626	2/3/7/13/31	4626	0
8M	4958	2/3/7/13/33	4958	0
9M	5271	2/3/7/12/26	4404	34
10M	5567	2/3/7/12/22	3850	256
12M	6120	2/3/7/12/19	3150	1995
14M	6628	2/3/7/11/18	3017	3886

## Wire Planning for Performance

[Otten-Brayton, DAC'98; Gosti et al., ICCAD'98]

- Monotonic wire plans for a functional network
  - ◆ Duplicating functionality may be needed
- Valid retiming
- Layer assignment
- Layout synthesis
  - ◆ Fixed delay
  - ◆ Cell generation and shape assignment

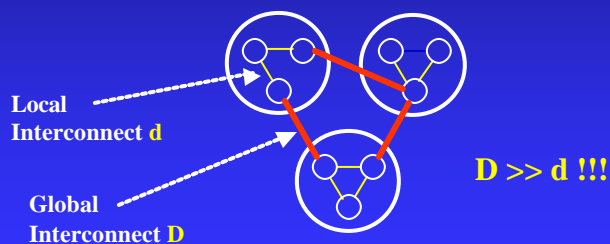
## Part V Outline

- Interconnect-Centric Design Flow
- Interconnect Performance Estimation
- Examples of Interconnect Planning
  - ◆ Problem formulation
  - ◆ Buffer block planning
  - ◆ Wire width planning
- System-Level Partitioning with Retiming
  - ◆ Hierarchical Performance-Driven Partitioning with retiming
- Concluding Remarks

## Motivation: Importance of Partitioning in Top-Down Chip-Assembly

### ■ Importance of Partitioning:

- ◆ Conventional view: enables divide-and-conquer
- ◆ DSM view: **defines global and local interconnects**
- ◆ Small blocks (50K-100K gates) can be synthesized reliably
- ◆ **Key is chip assembly considering global interconnects**



## Motivation: Importance of Retiming in Chip-Assembly

- Performance of global interconnects cannot support multi-gigahertz designs, even with aggressive optimization.

<b>Technology (um)</b>	<b>0.25</b>	<b>0.18</b>	<b>0.15</b>	<b>0.13</b>	<b>0.10</b>	<b>0.07</b>
<b>Intrinsic gate delay (ns)</b>	<b>0.071</b>	<b>0.051</b>	<b>0.049</b>	<b>0.045</b>	<b>0.039</b>	<b>0.022</b>
<b>1mm (ns)</b>	<b>0.059</b>	<b>0.049</b>	<b>0.051</b>	<b>0.044</b>	<b>0.052</b>	<b>0.042</b>
<b>2cm no-opt (ns)</b>	<b>2.589</b>	<b>2.48</b>	<b>2.65</b>	<b>2.62</b>	<b>3.73</b>	<b>4.67</b>
<b>2cm best-opt (ns)</b>	<b>0.895</b>	<b>0.793</b>	<b>0.77</b>	<b>0.7</b>	<b>0.77</b>	<b>0.672</b>

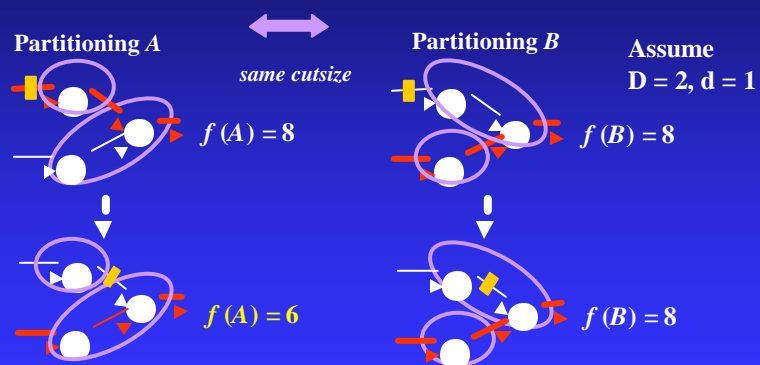
- Best-opt uses simultaneous buffer insertion, driver/buffer sizing, and wiresizing
- Reverse scaling of higher metal layers were not considered
- Source: [Cong97] SRC Working Papers <http://www.src.org/research/frontier.dgw>

## Motivation: Importance of Retiming in Chip-Assembly (Cont'd)

- Need multiple clock cycles to across the chip
- Requires new design and optimization techniques
  - ◆ Retiming and pipelining on global interconnects
  - ◆ New architectures based on only local interconnects?
  - ◆ Some degree of asynchronous designs?

## Motivation: Simultaneous Retiming during Partitioning

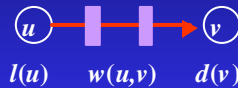
- Proper partitioning allows retiming to **hide** global interconnect delays.



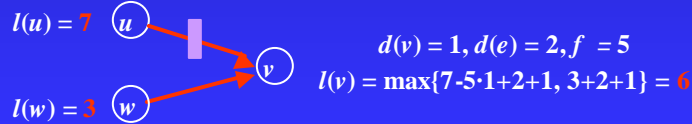
## Sequential Arrival Time (SAT)

### Definition [Pan et al, TCAD98]

- ◆  $l(v) = \text{max delay from PIs to } v \text{ after opt. retiming under a given clock period } f$
- ◆  $l(v) = \text{max}\{l(u) - f \cdot w(u,v) + d(u,v) + d(v)\}$



- ◆ Relation to retiming:  $r(v) = \lfloor l(v) / f \rfloor - 1$
- ◆ Theorem:  $P$  can be retimed to  $f + \text{max}\{d(e)\}$  iff  $l(\text{POs}) \leq f$



## Computation of SAT

### Single source longest path algorithm

- ◆ Edge lengths may be negative due to FFs
- ◆ Requires multiple iterations before convergence
- ◆ Complexity:  $O(n^2)$  in worst case but  $O(n)$  in practice

```

initialize  $l(v) = -\infty, l(\text{PI}) = 0;$ 
for ( $i = 1$  to  $|V|$ )
    DONE = false;
    visit each vertex  $v$ 
        for each fan-in  $u$  of  $v$ 
             $l'(v) = \text{max}\{l(u) - f \cdot w(u,v) + d(u,v) + d(v)\};$ 
            if ( $l(v) < l'(v)$ )
                 $l(v) = l'(v); \text{DONE} = \text{true};$ 
            if ( $v = \text{PO and } l(v) > f$ )
                return FAILURE;
    if (DONE = false)
        return SUCCESS;
    
```

## Simultaneous Partitioning with Retiming

-- Theory

- Node label: SAT at this node in optimally partitioned and retimed circuit w.r.t. a target clock period  $f$
- Theorem [Pan *et al*, TCAD'98]
  - ◆ Node labels can be computed in polynomial-time
  - ◆ If node label of some PO  $\in F$ , then  $F$  is not a feasible clock period
  - ◆ If node label of every PO  $\in F$ , then  $F + D - I$  is a feasible clock period
- Implication
  - ◆ quasi-optimal algorithm for partitioning with retiming
  - ◆ away from the optimal by at most  $D - I$

## Simultaneous Partitioning with Retiming

-- Theory (cont)

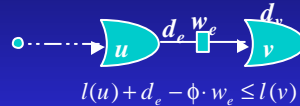
- Limitations of existing work [Pan *et al*, TCAD'98]
  - ◆ High space requirement:  $O(n^2)$
  - ◆ High time complexity:  $O(n(n+m) \log^2 n)$

## Simultaneous Partitioning with Retiming

-- A Scalable Solution [Cong-Li-Wu, DAC99]

### ■ Theory: revealed monotone property of node labels

- ◆ Captures the non-decreasing nature of sequential node arrival time from inputs to outputs



### ■ Algorithm:

- ◆ significant improvement of space and runtime over [Pan *et al*, TCAD'98]
- ◆ Significant reduction of candidate set for label update: a factor of  $O(\log n)$  reduction of runtime
- ◆ Efficient longest path computation --  $O(n)$  reduction on space & time complexity

## Simultaneous Partitioning with Retiming

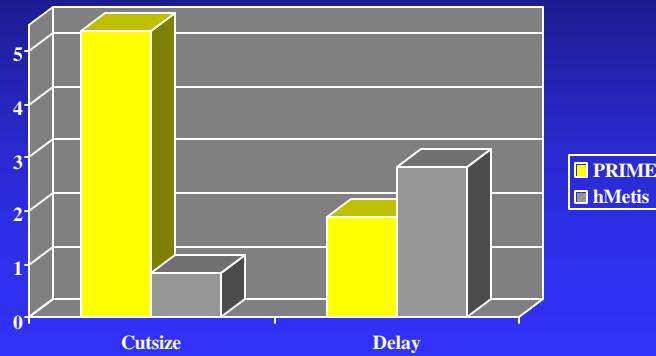
-- A Scalable Solution [Cong-Li-Wu, DAC99]

### ■ Results:

- ◆ A highly scalable solution to simultaneous partitioning and retiming (over 1000x speed-up for large designs)
- ◆ Still maintain quasi-optimality (at most  $D-1$  away from optimal clock period)

## Experimental Results of PRIME

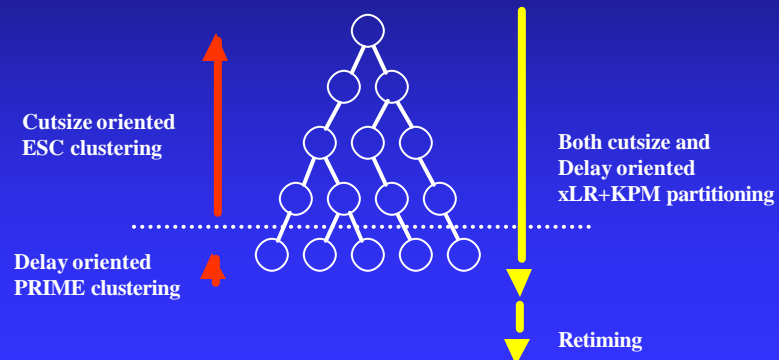
- 40% delay reduction vs. state-of-the-art algorithm hMetis [KA+97]
- Large cutsize: lack of consideration of cutsize reduction in PRIME



## Simultaneous Partitioning with Retiming:

-- A Scalable Solution With Good Performance/Cutsize Trade-off  
[Cong-Lim-Wu, DAC'2000]

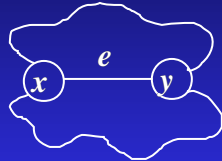
- A multi-level approach
  - ◆ Bottom-up multi-level clustering
  - ◆ Top-down multi-level partitioning + retiming



## Multi-level Clustering Using Edge Separability

[Cong and Lim, ASPDAC00]

- For  $(x, y) \in E$ , edge separability  $\psi(x, y) = \min \#$  edges needed to separate  $x$  and  $y$  (identical to  $x$ - $y$  mincut)



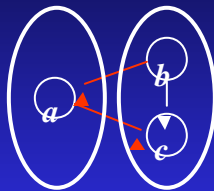
$$w(e) \leq q(e) \leq \psi(e)$$

- ESC (Edge Separability Based Clustering) Algorithm
  - ◆ Can compute a tight lower-bound  $q(e)$  of  $\psi(e)$  for *all* edges in  $O(n \log n)$  time using Maximum Spanning Forest [Nagamochi and Ibaraki, Algorithmica 1992]
  - ◆ Use  $q(e)$  for bottom-up multi-level clustering
  - ◆ Produce very good cutsizes, comparable to state-of-art hMetis [KA+97]

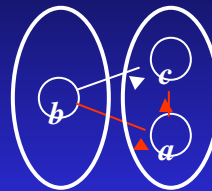
## Relaxed Acyclic Partitioning

[Cong and Lim, ASPDAC00]

- Motivation: avoid critical paths to be cut multiple times



Cyclic,  $f = 2D + 3d$

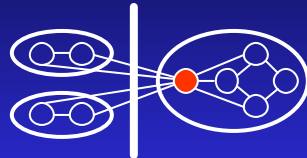


Acyclic,  $f = D + 3d$

- xLR Partitioning Algorithm
  - ◆ New gain function that discourages violation of acyclicity
  - ◆ Effective in improving delay while maintaining cutsizes

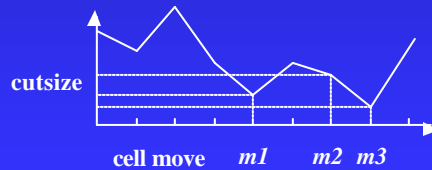
## Declustering and Refinement

- Declustering increases search space at each level



Cutsizes can be reduced from 4 to 3

- MRP (Multiple Rollback Point) scheme



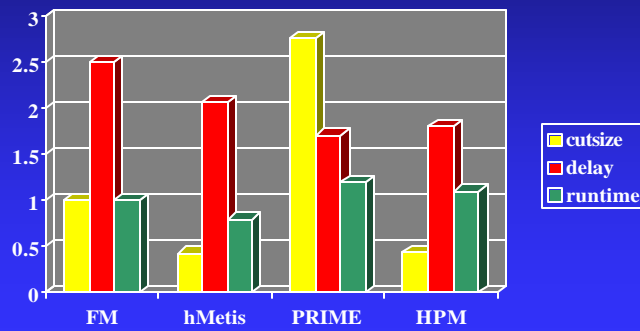
FM : keep *m3* only  
MRP : keep *m1*, *m2*, *m3* and pick min-delay sol.

## Summary of HPM Algorithm

- Multi-level approach
  - ◆ PRIME clustering guarantees quasi-optimal delay at bottom level
  - ◆ ESC clustering improves cutsizes while preserving quasi-optimal PRIME clusters
  - ◆ xLR partitioning and MRP scheme improve delay while maintaining cutsizes during declustering
  - ◆ Retiming further reduces delay

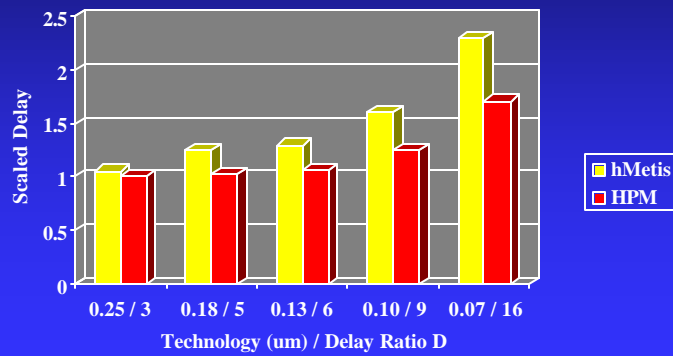
## HPM: Experimental Results

- Comparison among existing algorithms
  - FM [FM82], hMetis [KA+97], and PRIME [CLW99]



## HPM: Experimental Results

- D/d increases as technology further scales
  - Delay advantage increases as technology advances



## Concluding Remarks

- High-performance designs in DSM technologies need carefully interconnect planning
- Efficient interconnect performance estimation models (IPEMs) are important for interconnect planning
- Buffer block planning and wire width planning help to reduce complexity while achieving good performance
- Top-level partitioning defines global and local interconnects, and impacts performance significantly
- Retiming and pipelining over global interconnects are necessary for multi-gigahertz designs
- A clever combination of partitioning and retiming can hide (some) global interconnect delays

## Acknowledgments

- Thanks to Sung Lim, David Pan, and Xin Yuan at UCLA for their help with slides
- Thanks to SRC, MARCO/GSRC, and Intel Corp. for their supports of a number of research projects covered in this tutorial
- Updated slides in PDF file will be available at <http://cadlab.cs.ucla.edu/~cong>