Interconnect Design for Deep Submicron ICs

Jason Cong
(cong@cs.ucla.edu)
UCLA Computer Science Department
Los Angeles, CA 90095

Joint Work with L. He, K. Y. Khoo, C. K. Koh and Z. Pan

UCLA

ICCAD'97 Embedded Tutorial 1

Outline

1. Understandable to everyone
2. Understandable to intended audience
3. Understandable to experts only, such as the speaker
4. Understandable to nobody, including the speaker

Jason Cong 2 ICCAD'97 Embedded Tutorial

Real Outline

- Interconnect Trends and Challenges
- Delay Modeling
- Interconnect Layout Optimization
- Comparative Studies and Optimization Results
- Conclusions

National Technology Roadmap for Semiconductors (NTRS)

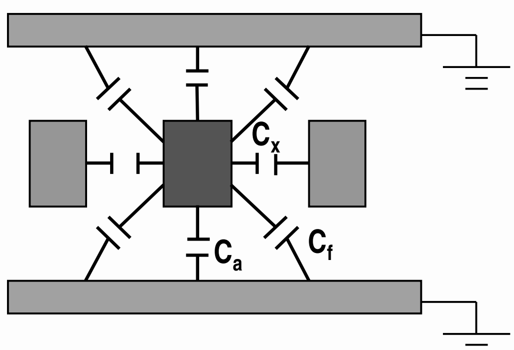
Technology (um)	*0.25*	*0.18*	*0.15*	*0.13*	*0.10*	*0.07*
Year	1997	1999	2001	2003	2006	2009
# transistors	11M	21M	40M	76M	200M	520M
On-Chip Clock (MHz)	750	1250	1500	2100	3500	6000
Area (mm^2)	300	340	385	430	520	620
Wiring Levels	6	6-7	7	7	7-8	8-9

Interconnect Parameters from NTRS

Technology (um)	***0.25***	***0.18***	***0.15***	***0.13***	***0.10***	***0.07***
Res. ρ (uΩ/cm)	3.3	2.2	2.2	2.2	2.2	1.8
Dielectric constant	3.55	2.75	2.25	1.75	1.75	1.5
Min. wire width	0.25	0.18	0.15	0.13	0.10	0.07
Min. wire spacing	0.34	0.24	0.21	0.17	0.14	0.10
Metal aspect ratio	1.8:1	1.8:1	2.0:1	2.1:1	2.4:1	2.7:1
Via aspect ratio	2.2:1	2.2:1	2.4:1	2.5:1	2.7:1	2.9:1
Vdd (V)	2.15	1.65	1.35	1.35	1.05	0.75

Jason Cong 5 ICCAD'97 Embedded Tutorial

Capacitance Computation



- **Capacitance extraction using a 3D field solver (FastCap)**

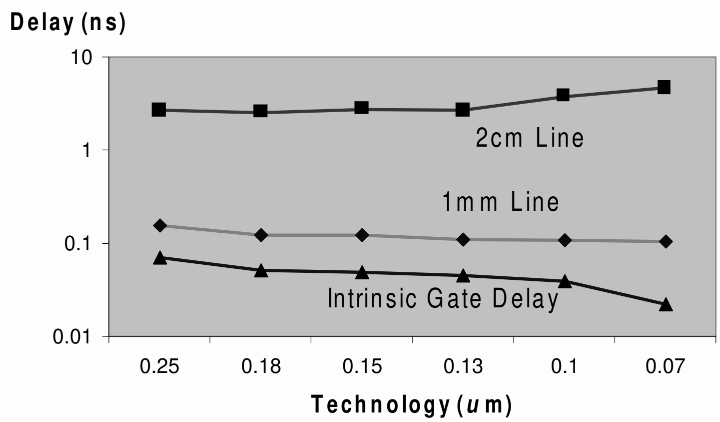
Jason Cong 6 ICCAD'97 Embedded Tutorial

Derived Interconnect & Device Parameters

Technology (um)		*0.25*	*0.18*	*0.15*	*0.13*	*0.10*	*0.07*
2X min. width & spacing	Ca (aF/um)	29.0	21.2	16.2	12.0	14.4	8.56
	Cf (aF/um)	41.8	30.2	24.8	18.3	14.1	14.8
	Cx(aF/um)	71.0	58.3	49.4	42.8	45.3	41.6
5X min. width & spacing	Ca (aF/um)	73.5	53.6	40.6	30.0	26.6	19.5
	Cf (aF/um)	63.5	47.3	38.4	28.5	28.2	23.6
	Cx(aF/um)	18.3	16.9	15.4	14.8	16.5	16.7
Buffer input cap. (fF)		0.17	0.12	0.11	0.085	0.070	0.042
Buffer Rd (x10KΩ)		1.71	1.86	2.26	2.25	2.39	2.42
Buffer intrin. delay (ps)		70.5	51.1	48.7	45.8	39.2	21.9

Jason Cong 7 ICCAD'97 Embedded Tutorial

Interconnect Performance Trend



No interconnect optimization (driver sizing only)

Jason Cong 8 ICCAD'97 Embedded Tutorial

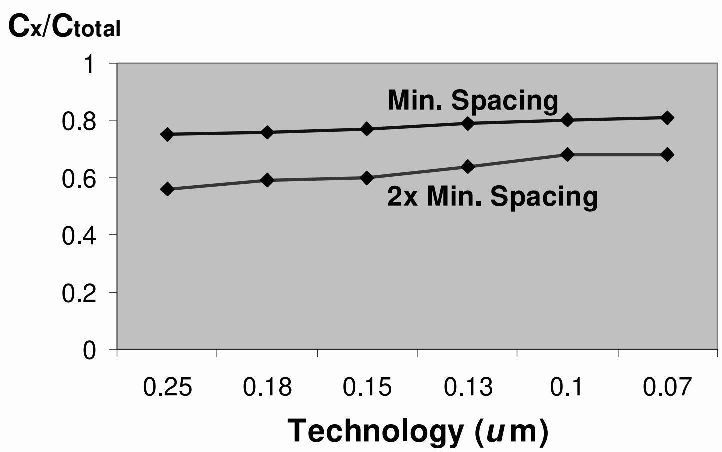
Interconnect Performance Trend

Technology (um)	*0.25*	*0.18*	*0.15*	*0.13*	*0.10*	*0.07*
2cm line delay (ns)	2.70	2.56	2.72	2.68	3.8	4.7
1mm line delay (ns)	0.155	0.124	0.124	0.109	0.108	0.105
Intrinsic gate delay (ns)	0.071	0.051	0.049	0.045	0.039	0.022

No interconnect optimization (driver sizing only)

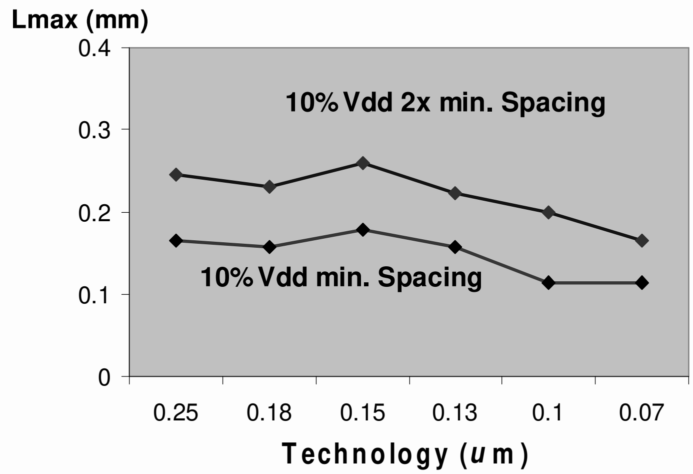
Jason Cong 9 ICCAD'97 Embedded Tutorial

Significance of Coupling Capacitance



Jason Cong 10 ICCAD'97 Embedded Tutorial

Coupling Noise



Jason Cong 11 ICCAD'97 Embedded Tutorial

Interconnect Complexity

Technology (um)	*0.25*	*0.18*	*0.13*	*0.10*	*0.07*
Length (m)	820	1,480	2,840	5,140	10,000
Wiring Levels	6	6-7	7	7-8	8-9

Jason Cong 12 ICCAD'97 Embedded Tutorial

Interconnect Complexity

Technology (um)	*0.25*	*0.18*	*0.13*	*0.10*	*0.07*
Length (m)	820	1,480	2,840	5,140	10,000
Wiring Levels	6	6-7	7	7-8	8-9
Opt. # buffers per net	few →				many
Opt. # wiresizes per net	few →				many
Opt. # buffers per chip	5K	25K	54K	230K	797K

- **Performance**
- **Signal reliability**
- **Electromigration**

Outline

- **Interconnect Trends and Challenges**
- **<u>Delay Modeling</u>**
- **Interconnect Layout Optimization**
- **Comparative Studies and Optimization Results**
- **Conclusions**

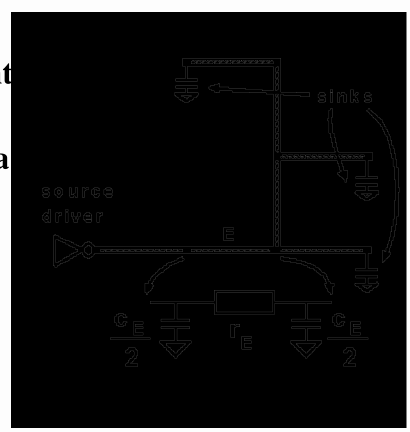
Delay Modeling

- Interconnect Modeling
- Driver Modeling
- Delay Computation

Jason Cong 15 ICCAD'97 Embedded Tutorial

Model Interconnects as RC Trees

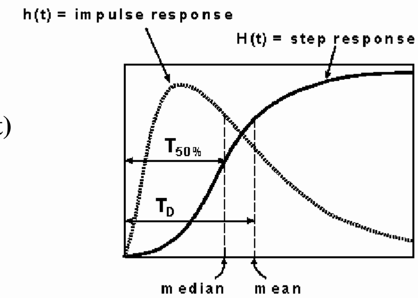
- Each wire maybe segment
  into several edges
- Each edge E modeled as a
  type or L-type circuit
- r_E = unit res. × length(E)
- c_E = unit cap. × length(E)



Jason Cong 16 ICCAD'97 Embedded Tutorial

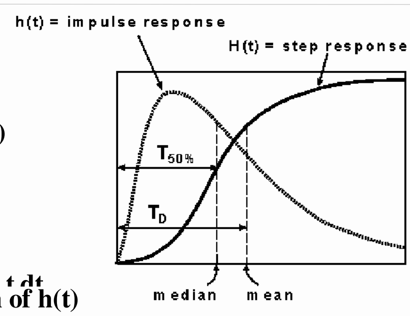
Elmore Delay for RC Trees

- **Definition**
  - **H(t) = step response**
  - **h(t) = impulse response**
    = rate of change of H(t)
  - **$T_{50\%}$ = median of h(t)**



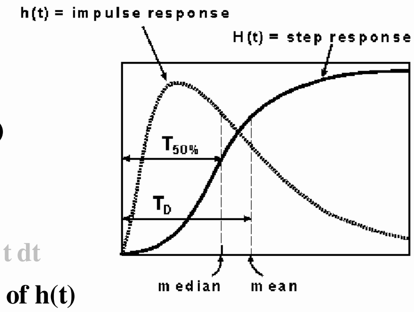
Elmore Delay for RC Trees

- **Definition**
  - **H(t) = step response**
  - **h(t) = impulse response**
    = rate of change of H(t)
  - **$T_{50\%}$ = median of h(t)**
- **Elmore Delay**
  - **T_D = mean of h(t) = $\int^{\infty} h(t)\, t\, dt$**
  - **Approximates the median of h(t) by the mean of h(t)**

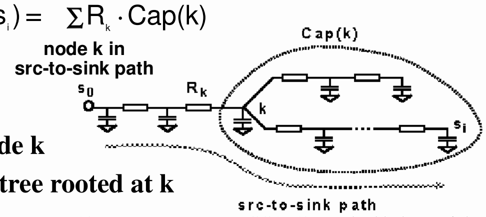


Elmore Delay for RC Trees

- **Definition**
  - **H(t) = step response**
  - **h(t) = impulse response**
    = rate of change of H(t)
  - **$T_{50\%}$ = median of h(t)**



- **Elmore Delay**
  - **T_D = mean of h(t) =** $\int_0^\infty h(t) \cdot t\,dt$
  - **Approximates the median of h(t) by the mean of h(t)**
- **Computation:** $T_D(s_0, s_i) = \sum R_k \cdot Cap(k)$



  - **s_0 = source**
  - **s_i = sink**
  - **R_k = resistance at node k**
  - **Cap(k) = total of subtree rooted at k**

Jason Cong 19 ICCAD'97 Embedded Tutorial

Properties of Elmore Delay

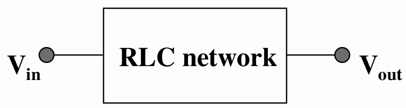
- **Advantages**
  - **Simple closed-form expression**
    - **Useful for interconnect optimization**
  - **Upper bound of 50% delay** [Gupta et al, 97]
    - **Actual delay asymptotically approaches Elmore delay as input signal rise time increases**
  - **High fidelity** [Boese et al., 93],[Cong-He,96]
    - **Good solutions under Elmore delay are good solutions under actual (SPICE) delay**
- **Disadvantages**
  - **Low accuracy, no waveform information**
  - **Inherently cannot handle inductance effect**

Jason Cong 20 ICCAD'97 Embedded Tutorial

Transfer Function & Moments

- **Laplace transform of impulse response h(t):**

$$H(s) = \int_0^\infty h(t) \cdot e^{-st}\, dt = \sum_{i=0}^\infty \frac{(-1)^i}{i!} \cdot s^i \cdot \int_0^\infty h(t) \cdot t^i\, dt$$



$$H(s) = \frac{V_{out}(s)}{V_{in}(s)}$$

- **Definition of i-th order moment:**

$$m_i = \frac{1}{i!} \cdot \int_0^\infty h(t) \cdot t^i\, dt \;\Rightarrow\; H(s) = m_0 - m_1 s + m_2 s^2 - \ldots$$

- **Elmore delay is first order moment**

Jason Cong 21 ICCAD'97 Embedded Tutorial

High Order Moments

- **Applicable to general RLC networks**
- **Used for waveform approximation (e.g. Asymptotic Waveform Evaluation (AWE))**
- **More accurate than Elmore delay**
- **Moment computation**
  - **Iterative DC analysis on transformed equivalent DC circuit** [Pillage-Rohrer, 90]
  - **Recursive computation based on tree traversal** [Ratzlaff-Pillage, 94] [Yu-Kuh, 95] [Kahng-Muddu, 95] [Cong-Koh, 97]

Jason Cong 22 ICCAD'97 Embedded Tutorial

Moment Matching Method

- **Asymptotic Waveform Evaluation** [Pillage-Rohrer, 90]
  - **Approximate $H(s)$ by q-pole transfer function**

    $$\hat{H}(s) = \frac{k_1}{s-p_1} + \frac{k_2}{s-p_2} + \dots + \frac{k_q}{s-p_q} \qquad k_i : \text{residues} \quad p_i : \text{poles}$$

  - **Match the initial boundary conditions and first 2q-1 moments of $H(s)$ and $\hat{H}(s)$ by Pade Approximation**
  - **Approximate time domain impulse response from residues and poles**

    $$\hat{h}(t) = k_i e^{p_1 t} + k_2 e^{p_2 t} + \dots + k_q e^{p_q t}$$

  - **Choice of q depends on accuracy requirement.**
    In practice, $q \le 5$ is frequently used for RC circuits

Jason Cong 23 ICCAD'97 Embedded Tutorial

Other Results on Interconnect Modeling and Analysis

- **Central moments [Gupta et al '95]**
- **PVL (Pade Approximation via the Lanczos Process) [Feldmann-Freund'94]**
- **MPVL [Feldmann-Freund'95]**
- **Congruence transformation [Kern-Wemple-Yang'95]**
- **Coordinate-transformed Arnoldi [Silveira, et al.'96]**
- **Dominant RC time constant model [Vandenberghe, et al'97]**
- **…...**

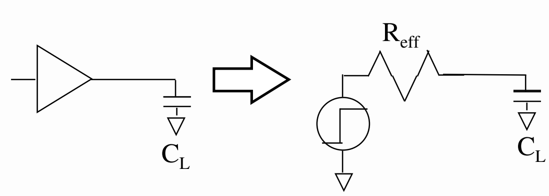
Jason Cong 24 ICCAD'97 Embedded Tutorial

Delay Modeling

- Interconnect Modeling
- Driver Modeling
- Delay Computation

Effective Resistance Model

- **Model the driver by an effective linear resistance**
  - Select an appropriate load C_L
  - Simulate and get 50% delay
  - Match the delay by that of an equivalent RC circuit ($R_{eff}C_L$)



- **Effective resistance may depend on input slope**

Driver Characterization

- **Table lookup model**
  - **Pre-characterize driver delay and transition by**
    - **input transition time t_t**
    - **total load capacitance C_L**
  - **typical entry: { t_t, C_L, (t_{delay}, $t_{rise/fall}$)}**
- **Table lookup and interpolation to obtain delay and transition time**
- **Can be very accurate**
- **Costly to create and store a multi-dimensional table**

Jason Cong 27 ICCAD'97 Embedded Tutorial

Driver Characterization (Cont'd)

- **k-factor equations for delay and output transition**

$$t_{delay} = (k_1 + k_2 \cdot C_L) \cdot t_t + k_3 \cdot C_L^3 + k_4 \cdot C_L + k_5$$

$$t_{r/f} = (k_1' + k_2' \cdot C_L) \cdot t_t + k_3' \cdot C_L^3 + k_4' \cdot C_L + k_5'$$

- **Determine coefficients k_i & k_i'**
  - **SPICE simulations for different combinations of input transition and load**
  - **Curve fitting by linear regression of least square fits**

Jason Cong 28 ICCAD'97 Embedded Tutorial

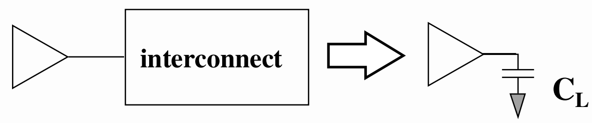
Delay Modeling

- Interconnect Modeling
- Driver Modeling
- <u>Delay Computation</u>

Delay Computation

- **Stage delay = delay from driver input A to loading gate C**
- **Two cases:**
  - **When interconnect resistance = 0**
  - **When interconnect = distributed RC**

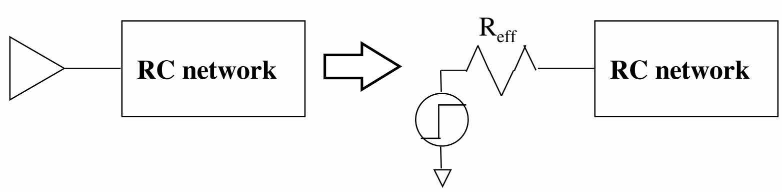
When Interconnect Resistance = 0



- C_L = total interconnect capacitance + sink capacitances
- Effective resistance model
  - Stage delay = $R_{eff}\,C_L$
- Pre-characterized driver model
  - Compute stage delay from table lookup or k-factor equations

When Interconnect = Distributed RC

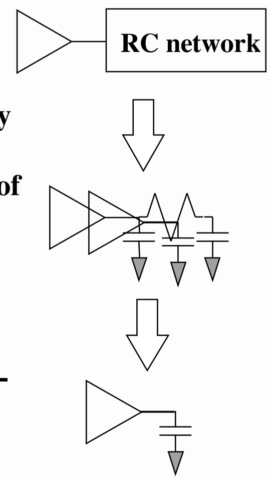
- Effective resistance model
  - Augment the RC network with the effective driver resistance



  - Compute stage delay using RC delay models

Distributed RC with Pre-characterized Driver Model

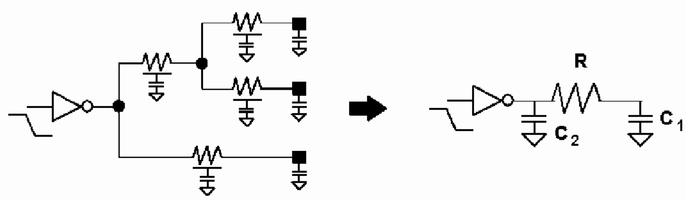
- Compute driver delay and interconnect delay separately
- Compute the effective capacitance seen by driver
  Difficulty: must capture shielding effect of interconnect resistance
- Solution:
  - Transform interconnect circuit into a π-model
  - Compute effective capacitance from π-model



Jason Cong 33 ICCAD'97 Embedded Tutorial

Construction of π-Model

[O'Brian-Savarino, 89]

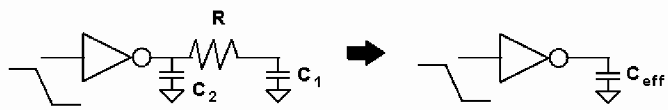


- Bottom-up computation of first three moments of driving point admittance y_1, y_2, y_3
  (i.e., moments of response current under voltage impulse at input)
- $C_1 = y_2^2 / y_3 \qquad C_1 = y_1 - C_1 \qquad R = -y_2^2 / y_2^3$

Jason Cong 34 ICCAD'97 Embedded Tutorial

Effective Capacitance Computation

[Qian-Pullela-Pileggi, 94]



- **Transform π-model to effective capacitance model**
- **Iterative computation of effective cap., t_{delay} & $t_{rise/fall}$**
  - **Start with C_{eff} = total interconnect + sink capacitances**
  - **Get t_{delay} & $t_{rise/fall}$ from pre-characterized driver model**
  - $C_{eff} = f(R, C_1, C_2, t_{delay}, t_{rise/fall})$

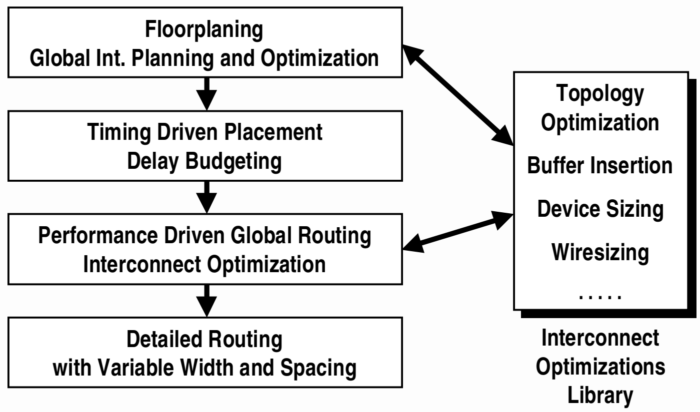
Jason Cong 35 ICCAD'97 Embedded Tutorial

Outline

- **Interconnect Trends and Challenges**
- **Delay Modeling**
- **<u>Interconnect Layout Optimization</u>**
- **Comparative Studies and Optimization Results**
- **Conclusions**

Jason Cong 36 ICCAD'97 Embedded Tutorial

Layout Design Flow for Deep Submicron ICs



Jason Cong 37 ICCAD'97 Embedded Tutorial

Interconnect Layout Optimization

- Wirelength Minimization
- Device Sizing
- Interconnect Topology Optimization
- Wiresizing Optimization
- Buffer Insertion
- Simultaneous Device and Interconnect Optimization

Jason Cong 38 ICCAD'97 Embedded Tutorial

Wirelength Minimization

- Through placement
  - Net-based approach
  - Path-based approach
- During global routing
  - Minimal wire length Steiner tree construction

Jason Cong 39 ICCAD'97 Embedded Tutorial

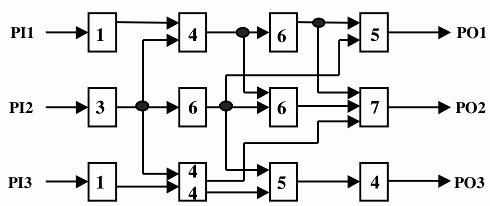
Net-based Placement

- Basic approach
  - Minimize $\sum_i w_i \cdot net\_length(i)$, where $w_i \propto \frac{1}{S_i}$

Jason Cong 40 ICCAD'97 Embedded Tutorial

Net-based Placement

- **Basic approach**
  - **Minimize** $\sum_i w_i \cdot net\_length(i)$, **where** $w_i \propto \frac{1}{S_i}$

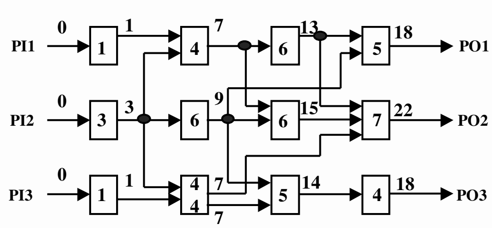


netlist with delay for each gate

Jason Cong 41 ICCAD'97 Embedded Tutorial

Net-based Placement

- **Basic approach**
  - **Minimize** $\sum_i w_i \cdot net\_length(i)$, **where** $w_i \propto \frac{1}{S_i}$

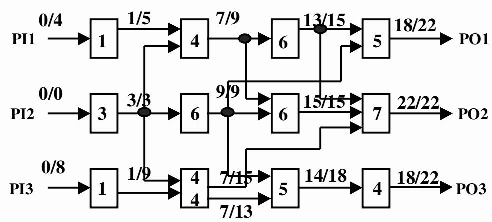


arrival times

Jason Cong 42 ICCAD'97 Embedded Tutorial

Net-based Placement

- **Basic approach**
  - **Minimize** $\sum_i w_i \cdot net\_length(i)$, **where** $w_i \propto \frac{1}{S_i}$

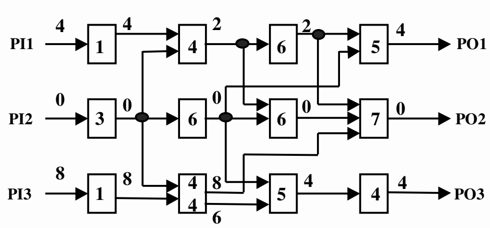


arrival time/required time

Jason Cong 43 ICCAD'97 Embedded Tutorial

Net-based Placement

- **Basic approach**
  - **Minimize** $\sum_i w_i \cdot net\_length(i)$, **where** $w_i \propto \frac{1}{S_i}$

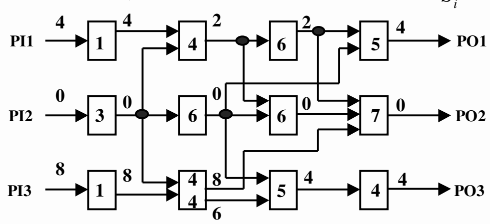


slack = required time - arrival time

Jason Cong 44 ICCAD'97 Embedded Tutorial

Net-based Placement

- **Basic approach**
  - **Minimize** $\sum_i w_i \cdot net\_length(i),$ **where** $w_i \propto \frac{1}{S_i}$

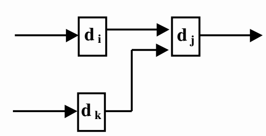


- **Delay budgeting algorithms (for computing S_i)**
  - **Zero slack algorithm** [Hauge-Nair-Yoffa'87]
  - **Limited bumping algorithm**[Frankle'92]

⋮

Path-based Placement

- **Basic approach**
  - **minimize** $\sum_i w_i \cdot net\_length(i),$
    subject to



……

$d_j \geq d_i + net\_delay(i, j) + gate\_delay(j)$

$d_j \geq d_k + net\_delay(k, j) + gate\_delay(j)$

……

$d_x \leq r_x, \forall x, x \in PO$

  - **Avoid path enumeration**
  - **Solved by mathematical programming (e.g. quadratic programming)**

Global Routing

- **Minimal wire length Steiner tree construction**
  - **Iterative addition of Steiner points**
  - **Merging edges of a minimum spanning tree (MST)**
  - **Iterative refinement of an MST**

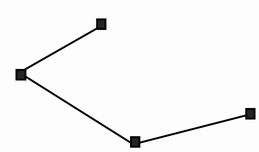
  ⋮

Jason Cong 47 ICCAD'97 Embedded Tutorial

Global Routing

- **Minimal wire length Steiner tree construction**
  - **<u>Iterative addition of Steiner points</u>**
  - **Merging edges of a minimum spanning tree (MST)**
  - **Iterative refinement of an MST**

  ⋮

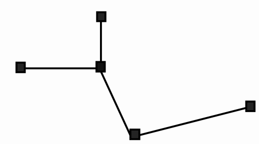


Jason Cong 48 ICCAD'97 Embedded Tutorial

Global Routing

- Minimal wire length Steiner tree construction
  - <u>Iterative addition of Steiner points</u>
  - Merging edges of a minimum spanning tree (MST)
  - Iterative refinement of an MST

  ⋮

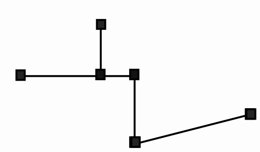


Jason Cong 49 ICCAD'97 Embedded Tutorial

Global Routing

- Minimal wire length Steiner tree construction
  - <u>Iterative addition of Steiner points</u>
  - Merging edges of a minimum spanning tree (MST)
  - Iterative refinement of an MST

  ⋮

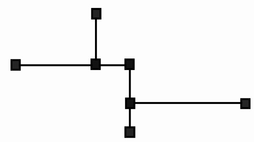


Jason Cong 50 ICCAD'97 Embedded Tutorial

Global Routing

- **Minimal wire length Steiner tree construction**
  - Iterative addition of Steiner points
  - Merging edges of a minimum spanning tree (MST)
  - Iterative refinement of an MST

  ⋮

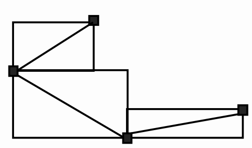


Jason Cong 51 ICCAD'97 Embedded Tutorial

Global Routing

- **Minimal wire length Steiner tree construction**
  - Iterative addition of Steiner points
  - Merging edges of a minimum spanning tree (MST)
  - Iterative refinement of an MST

  ⋮

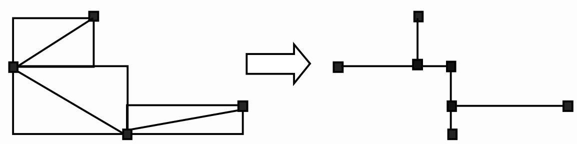


Jason Cong 52 ICCAD'97 Embedded Tutorial

Global Routing

- Minimal wire length Steiner tree construction
  - Iterative addition of Steiner points
  - <u>Merging edges of a minimum spanning tree (MST)</u>
  - Iterative refinement of an MST

    ⋮



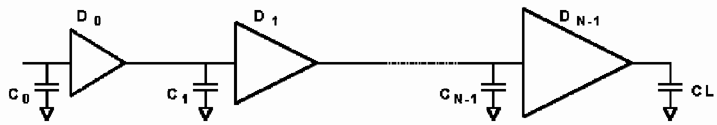
Jason Cong 53 ICCAD'97 Embedded Tutorial

Interconnect Layout Optimization

- Wirelength Minimization
- <u>Device Sizing</u>
- Interconnect Topology Optimization
- Wiresizing Optimization
- Buffer Insertion
- Simultaneous Device and Interconnect Optimization

Jason Cong 54 ICCAD'97 Embedded Tutorial

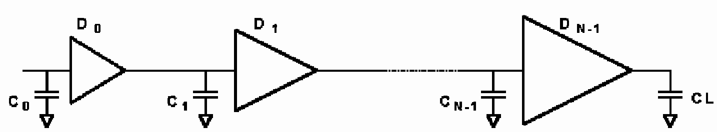
Device Sizing -- Driver Sizing



- **Given:** **A chain of cascaded drivers driving a load**
  Compute: **Optimal driver sizes to minimize delay, or total area under delay constraint**

Jason Cong 55 ICCAD'97 Embedded Tutorial

Device Sizing -- Driver Sizing



- **Given:** **A chain of cascaded drivers driving a load**
  Compute: **Optimal driver sizes to minimize delay, or total area under delay constraint**
- **Simple model:**
  - **Given N stages, optimal stage ratio** $f = (C_L / C_0)^{1/N}$
  - **Optimal stage number** $N = \ln C_L / \ln C_0, s.t. f = e$
- **More accurate model [Hedenstierna-Jeppson'87]**

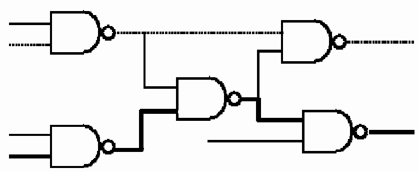
  $f = e^{(\alpha + f)/f}$

  $N = \ln(C_L / C_0) / \ln f$

Jason Cong 56 ICCAD'97 Embedded Tutorial

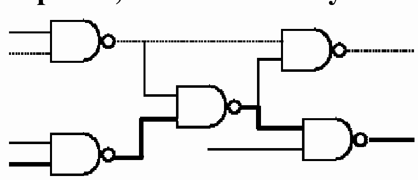
Device Sizing --Transistor/Gate Sizing

- **Given:** Logic network with or without cell library
  Find: Optimal size for each transistor/gate to minimize area or power, both under delay constraint



Device Sizing --Transistor/Gate Sizing

- **Given:** Logic network with or without cell library
  Find: Optimal size for each transistor/gate to minimize area or power, both under delay constraint



- **Static sizing:**
  - based on timing analysis and consider all paths at once (e.g [Fishburn-Dunlop'85])
- **Dynamic sizing:**
  - based on timing simulation and consider paths activated by given patterns [Conn et al'96]

Sensitivity-based Transistor Sizing

[Fishburn-Dunlop'85]]

- **Minimize** $area(X)$
  Subject to $Critical\_delay(X) \le T_{spec}$
  $X = \{x_1, \ldots, x_n\}$
- **Can be formulated as a posynomial program**
  - **Path_delay** $\sum a_{ij} \frac{r_i}{x_i} \bullet (c_j \bullet x_j)$ **is a posynomial function**

Jason Cong 59 ICCAD'97 Embedded Tutorial

Sensitivity-based Transistor Sizing

[Fishburn-Dunlop'85]]

- **Minimize** $area(X)$
  Subject to $Critical\_delay(X) \le T_{spec}$
  $X = \{x_1, \ldots, x_n\}$
- **Can be formulated as a posynomial program**
  - **Path_delay** $\sum a_{ij} \frac{r_i}{x_i} \bullet (c_j \bullet x_j)$ **is a posynomial function**
- **Begin with minimum size, iterate the following**
  - Compute sensitivities $\frac{\partial Path\_delay}{\partial x_i}$ for the current critical path
  - Size the transistor with the largest sensitivity

  Until $Critical\_delay(X) \le T_{spec}$

Jason Cong 60 ICCAD'97 Embedded Tutorial

Interconnect Layout Optimization

- Wirelength Minimization
- Device Sizing
- Interconnect Topology Optimization
- Wiresizing Optimization
- Buffer Insertion
- Simultaneous Device and Interconnect Optimization

Jason Cong 61 ICCAD'97 Embedded Tutorial

Interconnect Topology Optimization

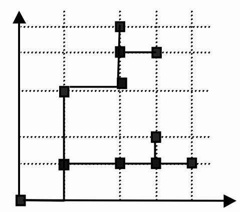
- resistance ratio = driver resistance / unit wire resistance
  - when resistance ratio is small, interconnect topology optimization is important
- Important metrics:
  - total wire-length
  - radius (longest source-sink path-length)
  - diameter (for multi-source nets)
- Optimal tree construction algorithms
  - BRBC (bounded-radius bounded-cost) algorithm
  - AHHK algorithm
  - A-tree algorithm
  - alphabetic -tree algorithm
  - p-tree algorithm, …...

Jason Cong 62 ICCAD'97 Embedded Tutorial

A-tree Algorithm

[Cong-Leung-Zhou'93]

- A-tree: Rectilinear Steiner arborescence (shortest path Steiner tree)



Jason Cong 63 ICCAD'97 Embedded Tutorial

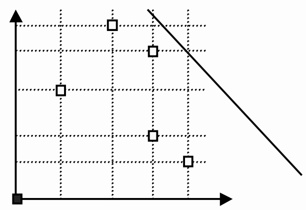
A-tree Algorithm

- A-tree: Rectilinear Steiner arborescence (shortest path Steiner tree)
- Basic A-tree algorithm
  - Start with a forest of n single-node A-trees, repeatedly combining two A-trees into a new one

Jason Cong 64 ICCAD'97 Embedded Tutorial

A-tree Algorithm

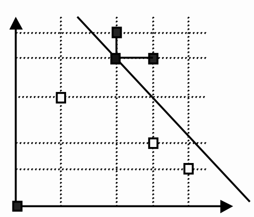
- A-tree: Rectilinear Steiner arborescence (shortest path Steiner tree)
- Basic A-tree algorithm
  - Start with a forest of n single-node A-trees, repeatedly combining two A-trees into a new one



Jason Cong 65 ICCAD'97 Embedded Tutorial

A-tree Algorithm

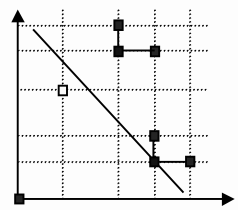
- A-tree: Rectilinear Steiner arborescence (shortest path Steiner tree)
- Basic A-tree algorithm
  - Start with a forest of n single-node A-trees, repeatedly combining two A-trees into a new one



Jason Cong 66 ICCAD'97 Embedded Tutorial

A-tree Algorithm

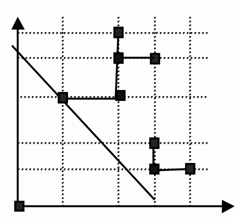
- A-tree: Rectilinear Steiner arborescence (shortest path Steiner tree)
- Basic A-tree algorithm
  - Start with a forest of n single-node A-trees, repeatedly combining two A-trees into a new one



Jason Cong 67 ICCAD'97 Embedded Tutorial

A-tree Algorithm

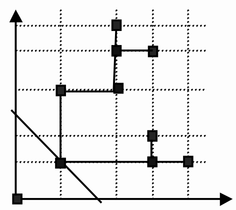
- A-tree: Rectilinear Steiner arborescence (shortest path Steiner tree)
- Basic A-tree algorithm
  - Start with a forest of n single-node A-trees, repeatedly combining two A-trees into a new one



Jason Cong 68 ICCAD'97 Embedded Tutorial

A-tree Algorithm

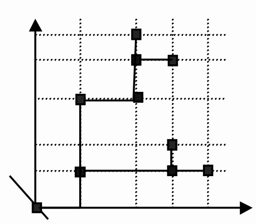
- A-tree: Rectilinear Steiner arborescence (shortest path Steiner tree)
- Basic A-tree algorithm
  - Start with a forest of n single-node A-trees, repeatedly combining two A-trees into a new one



Jason Cong 69 ICCAD'97 Embedded Tutorial

A-tree Algorithm

- A-tree: Rectilinear Steiner arborescence (shortest path Steiner tree)
- Basic A-tree algorithm
  - Start with a forest of n single-node A-trees, repeatedly combining two A-trees into a new one



Jason Cong 70 ICCAD'97 Embedded Tutorial

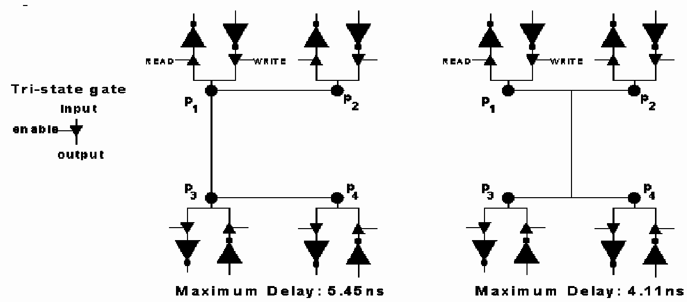
A-tree Algorithm

- **A-tree: Rectilinear Steiner arborescence (shortest path Steiner tree)**
- **Basic A-tree algorithm**
  - **Start with a forest of n single-node A-trees, repeatedly combining two A-trees into a new one**
  - **Improvement for near optimal/optimal solutions [Cong-Leung-Zhou'93]**
  - **Extension to routing graphs with weights/costs [Cong-Kahng_Leung'97]**

Performance Driven Multi-source Routing

[Cong-Madden'95]

- **Signal busses are examples of multi-source nets where each pin may act as a source, sink, or both**



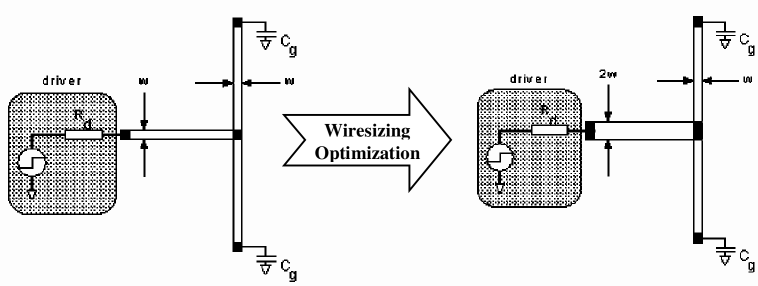
- **Minimum cost minimum diameter A-tree construction**
  - **Compute feasible region FR for root node**
  - **Construct A-trees at candidate root points in FR and select the best**

Interconnect Layout Optimization

- Wirelength Minimization
- Device Sizing
- Interconnect Topology Optimization
- <u>Wiresizing Optimization</u>
- Buffer Insertion
- Simultaneous Device and Interconnect Optimization

Discrete Wiresizing Optimization

- **Given:** A set of possible wire widths $\{ W_1, W_2, \ldots, W_r \}$
- **Find:** An optimal wire width assignment to
  1. Minimize the sum of weighted delays, or
  2. Minimize the maximum delay or maximize the min. slack at sinks



Wiresizing Optimization

- Discrete Wiresizing
  - Under Elmore delay model
    - single-source tree[Cong-Leung'93][Sapatnekar'94] [Lillis-Cheng-Lin'95]
    - multi-source tree [Cong-He'95]
    - non-tree [Xue-Kuh'95]
  - Under high-order delay model
    - RC-tree [Menezes et al.'94]
    - transmission line tree [Xue-Kuh-Yu'95]
- Optimal Wire Shaping
  - [Fishburn-Schevon'95][Chen-Zhou-Wong'96]

Jason Cong 75 ICCAD'97 Embedded Tutorial

Local Refinement and Dominance Relation for Wiresizing [Cong-Leung'93]

- Local Refinement of E

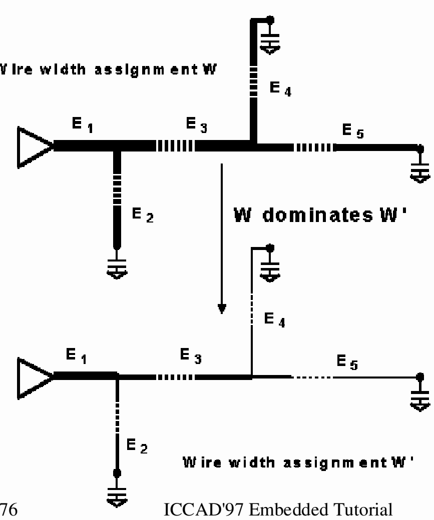
  Given wire width assignment W compute optimal wire width of E assuming other wire width fixed in W

- Dominance Relation

  For all E_j, $W(E_j) \geq W'(E_j)$

  $\Downarrow$

  W dominates W'



Jason Cong 76 ICCAD'97 Embedded Tutorial

Dominance Property [Cong-Leung'93]

- If solution W dominates optimal solution W*
  - W' = local refinement of W
    - Then, W' dominates W*
- If solution W is dominated by optimal solution W*
  - W' = local refinement of W
    - Then, W' is dominated by W*

Wiresizing Based on Dominance Property

- Lower bound computation

W_o = Min wire solution (dominated by opt. sol.)

Dominates ⇓

W_1 = Local-Refinement(W_0)

Dominates ⇓

W_2 = Local-Refinement(W_1)

⇓

- W_i dominated by opt. sol. $\Rightarrow$ a tight lower bound
- Upper bound can be computed similarly
- Extension to multi-source nets [Cong-He95]
- Also speedup by bundled refinement

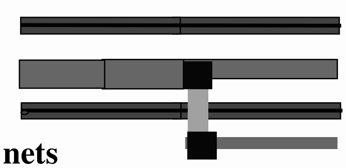
Wiresizing Based on High-Order Delay Models

- No explicit delay formula
- The sensitivity of moment is computed w.r.t. wire width
- To achieve target delays and slopes for RC tree [Menezes et al'94]
  - Invoke Levenburgh-Marquardt method to minimize mean square error between values of target and real moments
- To minimize maximum delay for transmission line tree [Xue-Kuh-Yu'96]
  - Iteratively increase/decrease the wire width with the largest sensitivity

Jason Cong 79 ICCAD'97 Embedded Tutorial

Global Interconnect Sizing and Spacing w/ Coupling Capacitance

[Cong-He-Koh-Pan'97]

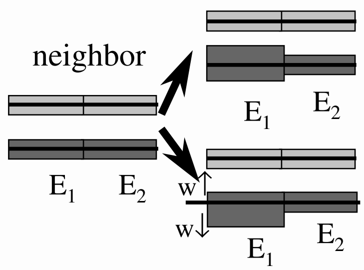


- Given:
  - Initial layout of multiple nets
  - Set of critical sinks and their criticalities
  - Capacitance models and design rules
- Output:
  - Sizing and spacing of each net for performance optimization

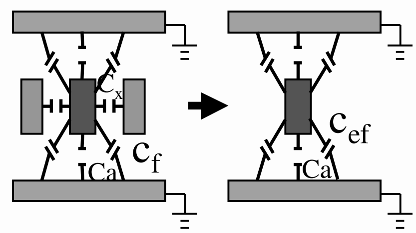
Jason Cong 80 ICCAD'97 Embedded Tutorial

Some Key Concepts for GISS

- Symmetric and Asymmetric wire sizing



- Effective-fringing capacitance



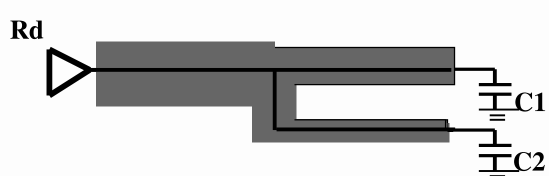
• **Reduce GISS to a set of single-net wiresizing problems with area and effective fringing cap.** $OWS(\overrightarrow{C_a}, \overrightarrow{C_{ef}})$

Jason Cong 81 ICCAD'97 Embedded Tutorial

Effective-Fringing Property

Larger C_{ef} leads to larger optimal wire-sizing, i.e.,

$$\overrightarrow{C_{ef}}' \geq \overrightarrow{C_{ef}} \Rightarrow OWS(C_a, \overrightarrow{C_{ef}}') \geq OWS(C_a, \overrightarrow{C_{ef}})$$



$$\overrightarrow{C_{ef}}' > \overrightarrow{C_{ef}}$$

Jason Cong

Bound Computation for Optimal GISS Solution

- Start with a min width solution
- Get effective fringing capacitance of each segment
- Compute optimal wiresizing solution for each net
- Re-compute effective fringing capacitance

⇩

Lower bound of opt. width of all nets!

(Due to the effective fringing property)

Jason Cong 83 ICCAD'97 Embedded Tutorial

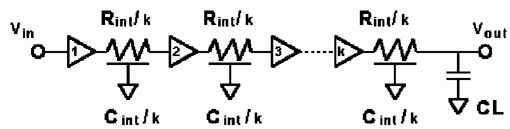
Interconnect Layout Optimization

- Wirelength Minimization
- Device Sizing
- Interconnect Topology Optimization
- Wiresizing Optimization
- <u>Buffer Insertion</u>
- Simultaneous Device and Interconnect Optimization

Jason Cong 84 ICCAD'97 Embedded Tutorial

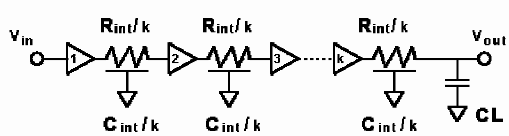
Buffer Insertion

- **Motivation**
  - **Reduce quadratic growth of $R_{int} \cdot C_{int}$ w.r.t. wirelength**
- **Classical approach**
  - **Insert equally-spaced buffers along a long wire**



Buffer Insertion

- **Motivation**
  - **Reduce quadratic growth of $R_{int} \cdot C_{int}$ w.r.t. wirelength**
- **Classical approach**
  - **Insert equally-spaced buffers along a long wire**



- **Buffer insertion for tree topology [van Ginneken'90]**
  - **Two-phase algorithm based on dynamic programming**
  - **Bottom-up to compute irredundant options at buffer candidate locations**
  - **Top-down to select the optimal locations**

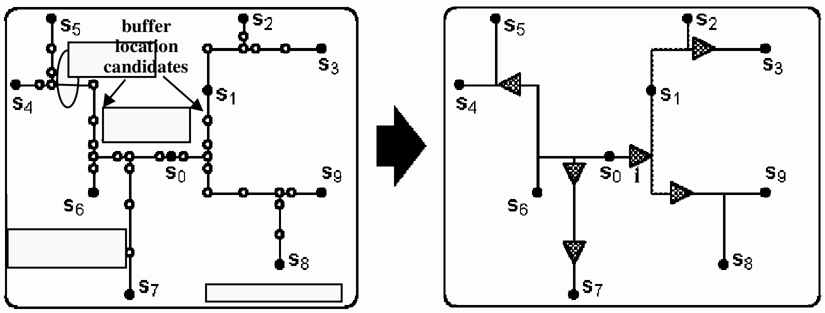
Buffer Insertion for Tree Interconnect

[van Ginneken, ISCAS'90]

- **Given topology, buffer types, and candidate buffer locations, insert buffers to**

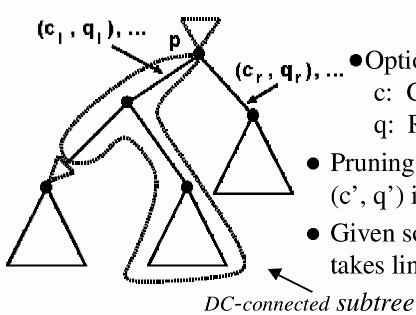
  (I) Minimize maximum sink delay

  (ii) Meet target delay for each sink



Optimal Buffer Insertion by Dynamic Programming

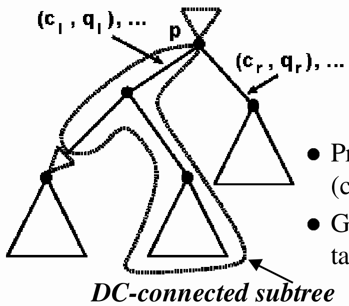
- **Bottom-up computation of irredundant set of options (c,q)'s at each buffer candidate location**



- Option (c,q),
  - c: Cap. of DC-connected subtree
  - q: Req. arrival time corresponding to c
- Pruning Rule: Given options (c,q), (c', q') (c', q') is redundant if $c' \geq c$ and $q' < q$
- Given sorted option lists for two sub-trees, it takes linear time to merge them at p.

Optimal Buffer Insertion by Dynamic Programming

- **Bottom-up computation of irredundant set of options (c,q)'s at each buffer candidate location**



- a possible solution = option (c,q):
  - c: Cap. of DC-connected subtree
  - q: Req. arrival time corresponding to c
- Pruning Rule: Given options (c,q), (c', q') (c', q') is redundant if $c' \geq c$ and $q' < q$
- Given sorted option lists for two sub-trees, it takes linear time to merge them at p.

- **Top-down selection of optimal buffer types and buffer locations**

Jason Cong 89 ICCAD'97 Embedded Tutorial

Interconnect Layout Optimization

- **Wirelength Minimization**
- **Device Sizing**
- **Interconnect Topology Optimization**
- **Wiresizing Optimization**
- **Buffer Insertion**
- **Simultaneous Device and Interconnect Optimization**

Jason Cong 90 ICCAD'97 Embedded Tutorial

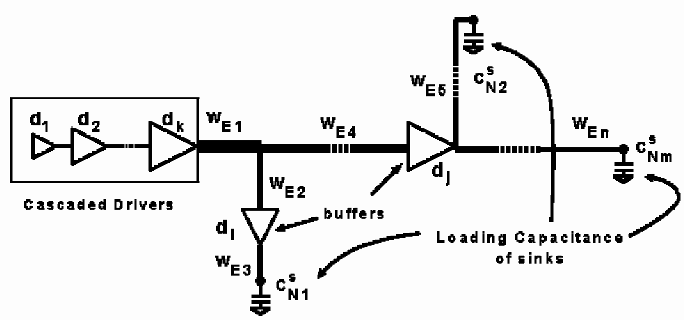
Simultaneous Buffer and Interconnect Optimization

- Simultaneous device and wiresizing
  - [Cong-Koh'94]
  - [Menezes-Baldick-Pileggi'95]
  - [Cong-He'96]
  - [Chen-Chang-Wong'96]
  - [Cong-Koh-Leung'96] …
- Buffer insertion with wiresizing
  - [Lillis-Cheng-Lin'95]
  - [Chu-Wong'97] …
- Simultaneous buffered tree construction with wiresizing
  - [Okamoto-Cong'96]

Jason Cong 91 ICCAD'97 Embedded Tutorial

Simultaneous Driver/Buffer and Wiresizing (SD/BWS) for Performance & Power Optimization

[Cong-Koh'94] [Cong-Koh-Leung'96]



Given:	**A buffered tree T with sinks and n buffers**
Determine:	**D = Driver sizes + Buffer sizes, W = Wire widths**
Minimize:	**1. Performance Measure = weighted-sum of sink delays**
	2. $\alpha \times$ Power + (1 - α) $\times$ Performance Measure

Jason Cong 92 ICCAD'97 Embedded Tutorial

Optimal SD/BWS Algorithm

- **Dominance (D, W) and (D', W') [Cong-Koh-Leung'96]**
  - **If W = opt-WS(D) and W' = opt-WS(D')**
    D dominates D' ⇒ W dominates W'
  - **If D = opt-BS(W) and D' = opt-BS(W')**
    W dominates W' ⇒ D dominates D'

Optimal SD/BWS Algorithm

- **Dominance (D, W) and (D', W') [Cong-Koh-Leung'96]**
  - **If W = opt-WS(D) and W' = opt-WS(D')**
    D dominates D' ⇒ W dominates W'
  - **If D = opt-BS(W) and D' = opt-BS(W')**
    W dominates W' ⇒ D dominates D'
- **Interleave buffer and wire sizing for bound computation**
  - **Lower bound of SBWS optimal solution**

Dominate

W_0 = Min. width solution (dominated by opt. sol.)
D_0 = Opt-BS(W_0)
W_1 = Opt-WS(D_0)
D_1 = Opt-BS(W_1)

- **Bottom-up dynamic programming if bounds do not meet**

Optimal SD/BWS Algorithm

- **Dominance (D, W) and (D', W') [Cong-Koh-Leung'96]**
  - **If W = opt-WS(D) and W' = opt-WS(D')**
    D dominates D' $\Rightarrow$ W dominates W'
  - **If D = opt-BS(W) and D' = opt-BS(W')**
    W dominates W' $\Rightarrow$ D dominates D'
- **Interleave buffer and wire sizing for bound computation**
  - **Lower bound of SBWS optimal solution**

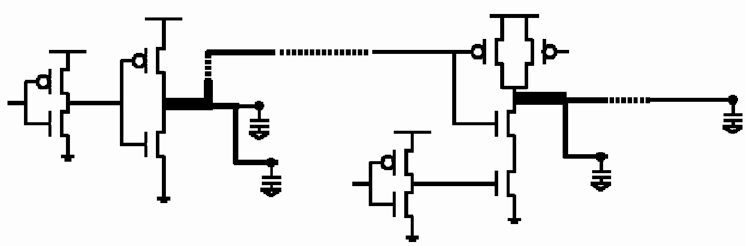
Dominate

W_0 = Min. width solution (dominated by opt. sol.)
D_0 = Opt-BS(W_0)
W_1 = Opt-WS(D_0)
D_1 = Opt-BS(W_1)

- **Bottom-up dynamic programming if bounds do not meet**
- **CH-posynomial based approach in [Cong-He'96]**
  - **General netlist of devices and interconnects**
  - **Table-based models for device delay and interconnect coupling**

Simultaneous Transistor and Interconnect Sizing

[Cong-He'96]



Given:	**Initial layout design for multiple nets,**
	Table-based models for device delay and interconnect coupling capacitances
Determine:	**Discrete sizes for transistors/wires**
Minimize:	**α Delay + β Power + γ Area**

Dominance Property for CH-Posynomial Programs

- **Minimize**

$$f(X) = \sum_{p=0}^{m} \sum_{q=0}^{m} \sum_{i=1}^{n} \sum_{j=1,\, j \neq i}^{n} \left(\frac{a_{pi}(X)}{x_i^p}\right) \cdot \left(b_{qj}(X) \cdot x_j^q\right)$$

  - **Simple CH-posynomial program when a_{pi} and b_{qj} are constants**
  - **General CH-posynomial program when a_{pi} and b_{qj} are functions of X**

Jason Cong 97 ICCAD'97 Embedded Tutorial

Dominance Property for CH-Posynomial Programs

- **Minimize**

$$f(X) = \sum_{p=0}^{m} \sum_{q=0}^{m} \sum_{i=1}^{n} \sum_{j=1,\, j \neq i}^{n} \left(\frac{a_{pi}(X)}{x_i^p}\right) \cdot \left(b_{qj}(X) \cdot x_j^q\right)$$

  - **Simple CH-posynomial program when a_{pi} and b_{qj} are constants**
  - **General CH-posynomial program when a_{pi} and b_{qj} are functions of X**
- **Dominance property holds for**
  - **Simple CH-posynomial program w.r.t. local refinement**
  - **General CH-posynomial program w.r.t. extended local refinement**

Jason Cong 98 ICCAD'97 Embedded Tutorial

Dominance Property for CH-Posynomial Programs

- Minimize

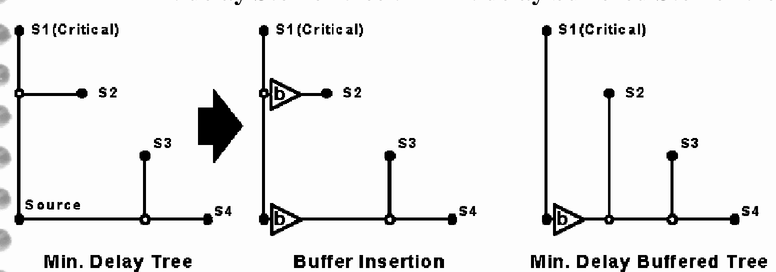
$$f(X) = \sum_{p=0}^{m} \sum_{q=0}^{m} \sum_{i=1}^{n} \sum_{j=1,\, j \neq i}^{n} \left(\frac{a_{pi}(X)}{x_i^p}\right) \cdot \left(b_{qj}(X) \cdot x_j^q\right)$$

  - Simple CH-posynomial program when a_{pi} and b_{qj} are constants
  - General CH-posynomial program when a_{pi} and b_{qj} are functions of X
- Dominance property holds for
  - Simple CH-posynomial program w.r.t. local refinement
  - General CH-posynomial program w.r.t. extended local refinement
- STIS is a general CH-posynomial program
  - Table-based models for device delay and interconnect coupling
  - Tight lower and upper bounds are computed

Wiresized Buffered Tree Construction

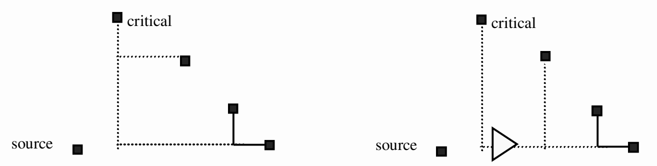
[Cong-Okamoto'96]

- Two-step approach vs. unified single-step approach
  - Buffer insertion followed by tree construction
    Difficulty of wiring delay and routability estimation in early stage
  - Tree construction followed by buffer insertion
    Min. delay Steiner tree != Min. delay buffered Steiner tree



Wiresized Buffered A-tree Algorithm (WBA-tree)

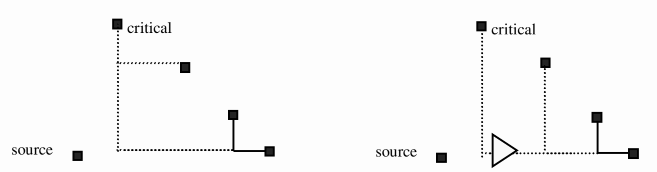
- **Bottom-up A-tree topology construction**
  - **Iterative merging of Tv and Tw with max. merging cost (*mcost*)**
  - ***mcost* measures the trade-off between:**
    - **maximization of required arrival time at source**
    - **minimization of wire/buffer capacitance**



  - **A set of options computed similar to [van Ginneken’90]**

Wiresized Buffered A-tree Algorithm (WBA-tree)

- **Bottom-up A-tree topology construction**
  - **Iterative merging of two subtrees with max. merging weight(*mweight*)**
  - ***mweight* measures the trade-off between:**
    - **maximization of wirelength reduction**
    - **maximization of required arrival time at source**
  - **A set of options computed similar to [van Ginneken’90]**



- **Top-down optimal selection of buffer types and locations**

Outline

- Interconnect Trends and Challenges
- Delay Modeling
- Interconnect Layout Optimization
- <u>Comparative Studies and Optimization Results</u>
- Conclusions

Jason Cong 103 ICCAD'97 Embedded Tutorial

Optimization Results and Comparative Studies

- Impact on Future Technology Generation
- Algorithm Comparisons
- Global Interconnect Sizing and Spacing (GISS) with Coupling Capacitance

Jason Cong 104 ICCAD'97 Embedded Tutorial

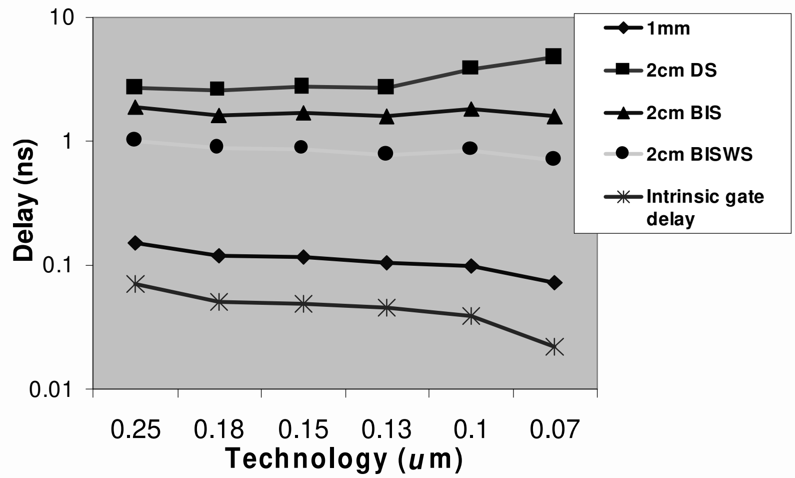
Interconnect Parameters from NTRS

Technology (um)	*0.25*	*0.18*	*0.15*	*0.13*	*0.10*	*0.07*
Res. ρ (uΩ/cm)	3.3	2.2	2.2	2.2	2.2	1.8
Dielectric constant	3.55	2.75	2.25	1.75	1.75	1.5
Min. wire width	0.25	0.18	0.15	0.13	0.10	0.07
Min. wire spacing	0.34	0.24	0.21	0.17	0.14	0.10
Metal aspect ratio	1.8:1	1.8:1	2.0:1	2.1:1	2.4:1	2.7:1
Via aspect ratio	2.2:1	2.2:1	2.4:1	2.5:1	2.7:1	2.9:1
Vdd (V)	2.15	1.65	1.35	1.35	1.05	0.75

Derived Interconnect & Device Parameters

Technology (um)		*0.25*	*0.18*	*0.15*	*0.13*	*0.10*	*0.07*
2X min. width & spacing	Ca (aF/um)	29.0	21.2	16.2	12.0	14.4	8.56
	Cf (aF/um)	41.8	30.2	24.8	18.3	14.1	14.8
	Cx(aF/um)	71.0	58.3	49.4	42.8	45.3	41.6
5X min. width & spacing	Ca (aF/um)	73.5	53.6	40.6	30.0	26.6	19.5
	Cf (aF/um)	63.5	47.3	38.4	28.5	28.2	23.6
	Cx(aF/um)	18.3	16.9	15.4	14.8	16.5	16.7
Buffer input cap. (fF)		0.17	0.12	0.11	0.085	0.070	0.042
Buffer Rd (x10KΩ)		1.71	1.86	2.26	2.25	2.39	2.42
Buffer intrin. delay (ps)		70.5	51.1	48.7	45.8	39.2	21.9

Impact of Interconnect Optimization on Future Technology Generations



Jason Cong 107 ICCAD'97 Embedded Tutorial

Impact of Interconnect Optimization on Future Technology Generations

Technology (um)	*0.25*	*0.18*	*0.15*	*0.13*	*0.10*	*0.07*
1mm (ns)	0.151	0.118	0.117	0.104	0.097	0.072
2cm DS (ns)	2.7	2.56	2.72	2.68	3.8	4.7
2cm BIS (ns)	1.89	1.63	1.69	1.58	1.8	1.58
2cm BISWS (ns)	1.01	0.87	0.86	0.77	0.84	0.7

Jason Cong 108 ICCAD'97 Embedded Tutorial

Impact on Future Technology

	Technology (um)	0.25	0.18	0.15	0.13	0.10	0.07
1 mm	Delay (ns)	0.15	0.12	0.12	0.10	0.097	0.072
	Number of buffers	1	1	1	1	1	1
	Avg. buffer size	94	102	96	99	114	144
	Avg. wire width	2.44	2.89	3.00	3.61	4.20	6.71
	% wire sized	85%	88%	90%	93%	94%	98%
2 cm	Delay (ns)	1.01	0.87	0.86	0.77	0.83	0.70
	Number of buffers	4	5	5	5	6	9
	Avg. buffer size	200	200	200	200	200	200
	Avg. wire width	6.52	6.61	6.93	7.31	7.70	8.00
	% wire sized	98.6%	98.9%	99.1%	99.3%	99.5%	99.8%

Jason Cong 109 ICCAD'97 Embedded Tutorial

Comparison of Several Interconnect Optimization Algorithms

- **T+B+W:Topology (T), followed by optimal buffer insertion and sizing B (B=10) then followed by optimal wire sizing (W=18)**
- **TB+BW: Simultaneous T and B (B=3), followed by simultaneous buffer and wire sizing (BW) with B=40, W=18**
- **Tbw+BW: Simultaneous TBW with small number of B=3 and W=3, then followed by BW as above**
- **TBW: Simultaneous TBW with larger number of B=10 and W=8**
- **Provided by the UCLA TRIO (Tree, Repeater, & Interconnect Optimization) package**

Jason Cong 110 ICCAD'97 Embedded Tutorial

Comparison of Optimization Results by Different Algorithms

		Algorithms			
		T+B+W	**TB+BW**	**Tbw+BW**	**TBW**
5-pin nets	**Delay (nS)**	0.40	0.39	0.35	0.34
		0.47	0.48	0.38	0.38
		0.42	0.41	0.36	0.35
	CPU (S)	0.1	0.1	1.4	15
10-pin nets	**Delay (nS)**	0.42	0.37	0.34	0.33
		0.56	0.56	0.44	0.44
		0.47	0.45	0.38	0.38
	CPU (S)	0.8	1.0	6.4	76
20-pin nets	**Delay (nS)**	0.45	0.43	0.38	0.39
		0.54	0.48	0.42	0.41
		0.46	0.43	0.38	0.38
	CPU (S)	1.6	4.0	27.6	350

Jason Cong 111 ICCAD'97 Embedded Tutorial

Global Interconnect Sizing and Spacing

Center spacing	*Average Delays(ns)*		
	MIN	OWS	GISS
2 x pitch	1.75	1.45 (-17%)	0.82 (-53%)
3 x pitch	1.51	0.83 (-45%)	0.56 (-63%)
4 x pitch	1.29	0.66 (-49%)	0.45 (-65%)
5 x pitch	1.21	0.60 (-50%)	0.42 (-65%)
6 x pitch	1.16	0.56 (-52%)	0.38 (-67%)

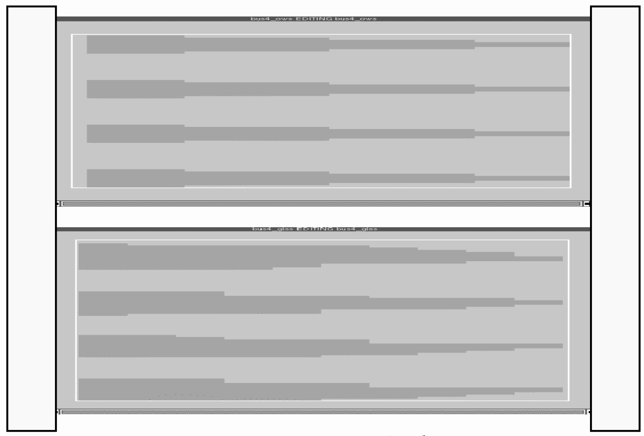
- **16-bit bus structure of 10 mm length**

Jason Cong 112 ICCAD'97 Embedded Tutorial

Global Interconnect Sizing and Spacing

OWS

GISS



Snapshot of upper four bus lines, 4 x *pitch* center spacing

Conclusions

- **Interconnect-driven design is an important methodology**
- **Interconnect design needs to be considered at every level of design process**
- **Interconnect layout optimization has a significant impact on circuit performance**
- **Coupling capacitance should be considered during performance optimization**