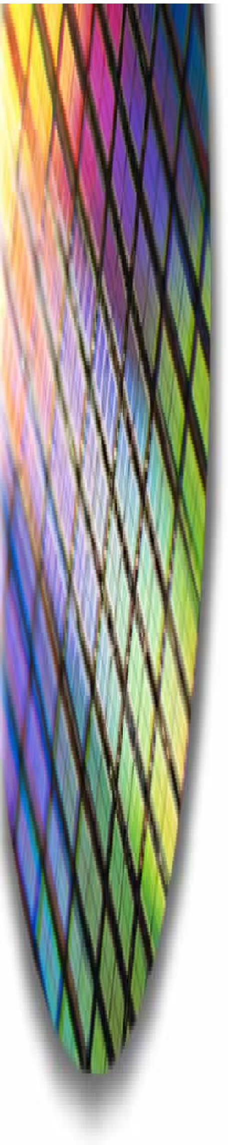
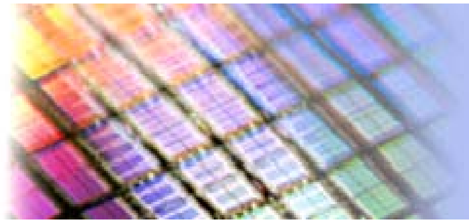


Variation Tolerant Buffered Clock Network Synthesis with Cross Links

Anand Rajaram [†, ‡] David Z. Pan [†]
[†] Dept. of ECE, UT-Austin
[‡] Texas Instruments, Dallas

Sponsored by SRC and IBM Faculty Award

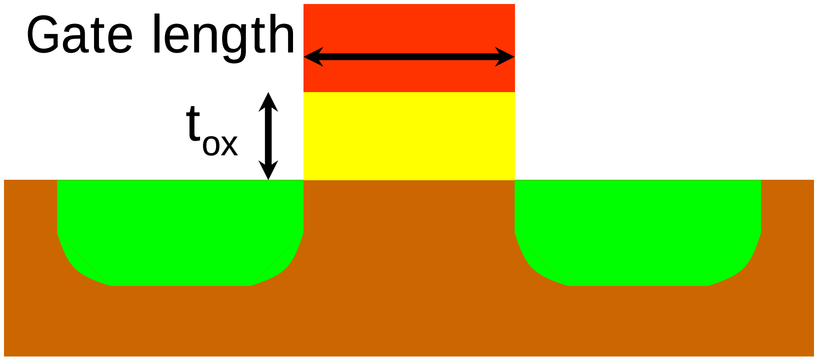


Presentation Outline

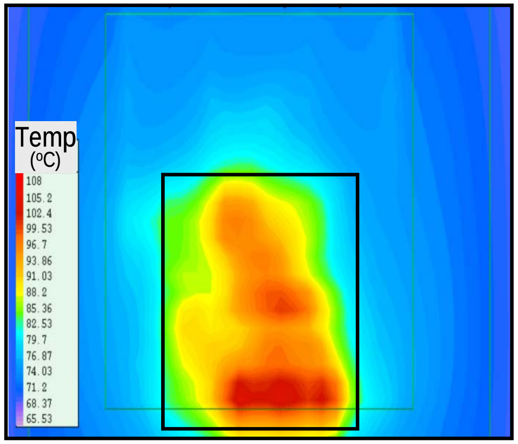
- Introduction
- Link Insertion and Challenges for Buffered Clock Trees
- Linked Buffered Clock Tree Synthesis
- Experimental Results
- Conclusions

Clock Network

- Stringent skew budget for multi-gHz designs
- Global in nature (span the entire chip)
- Skew is very sensitive to variations
  - Manufacturing process variations (P)
  - Supply voltage variations (V)
  - Temperature variations (T)
- => Variation-tolerant clock network

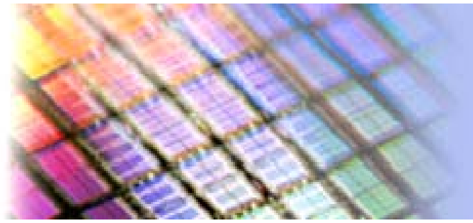


Gate variations



(source: Intel)

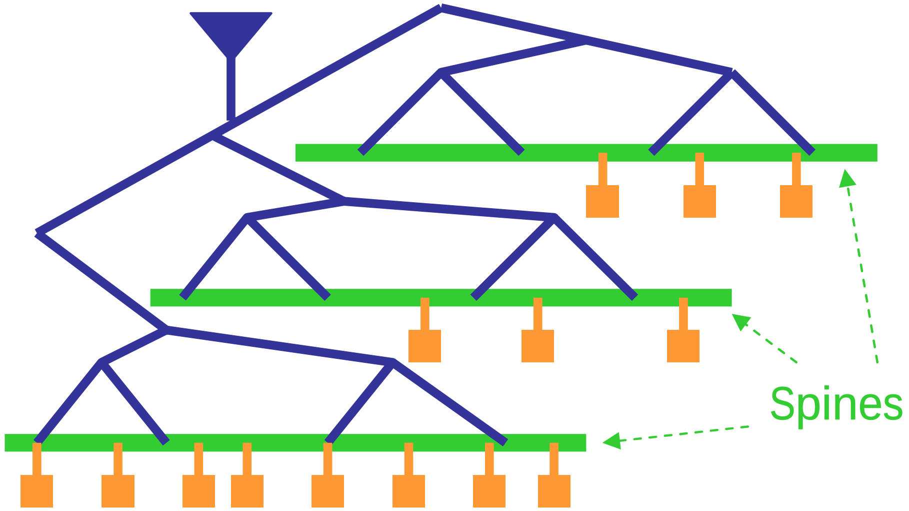
Temp variations



Approaches for Reducing Skew Variability

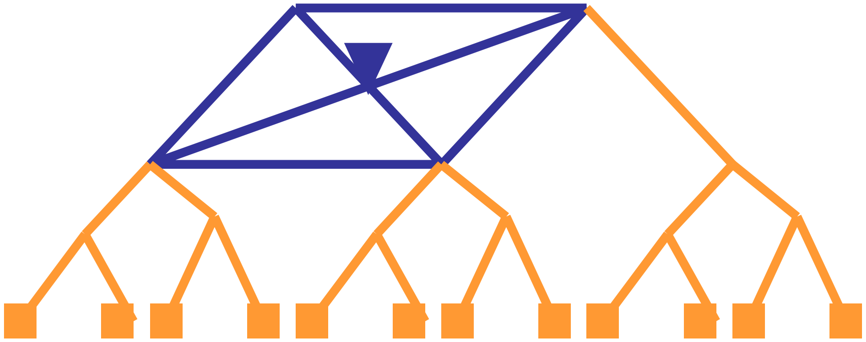
- Buffer & wire sizing [Pullela et al., DAC’93; Chung et al., ICCAD’94; Wang et al., ISPD’04]
- Variation aware routing [Lin et al., ICCAD’94; Lu et al., ISPD’03; Padmanabhan+, ISPD’06]
- Temperature aware clock optimization [Cho+, ICCAD’05]
- Non-tree clock network
  - McCoy+, ETC’94; Xue et al., ICCAD’95; Vandenberghe et al., ICCAD’97; Kurd et. al. JSSC’01; Su et. al. ICCAD’01; Restle et al. JSSC’01
  - Link based non-tree clock networks: Rajaram et al., DAC’04, ISPD’05, ISQED’06; Venkataraman+, ICCAD’05

Non-tree: Spine & Mesh

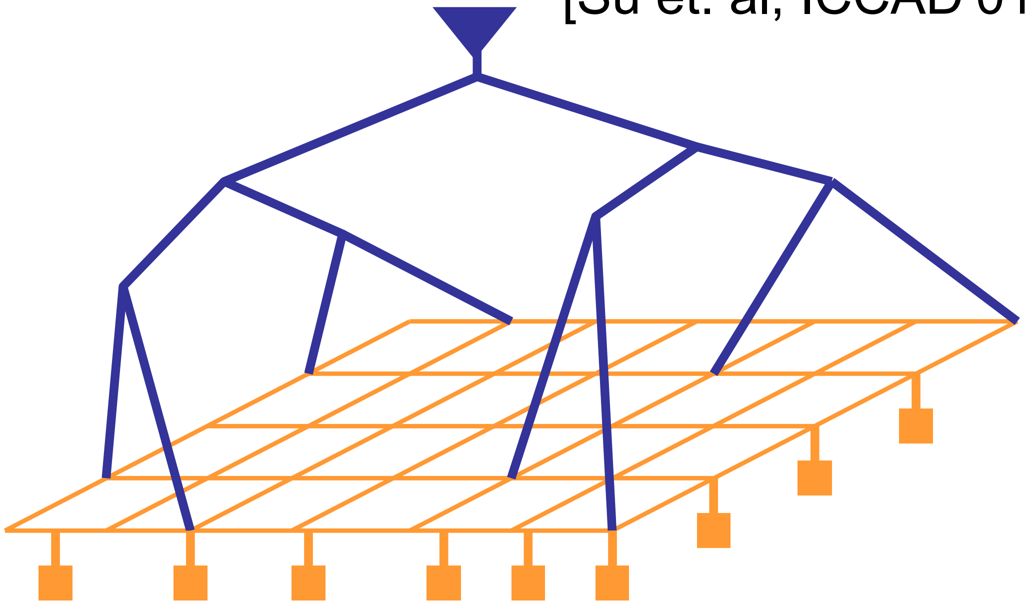


Applied in *Pentium* processor

[Kurd et. al. JSSC'01]



[Su et. al, ICCAD'01]



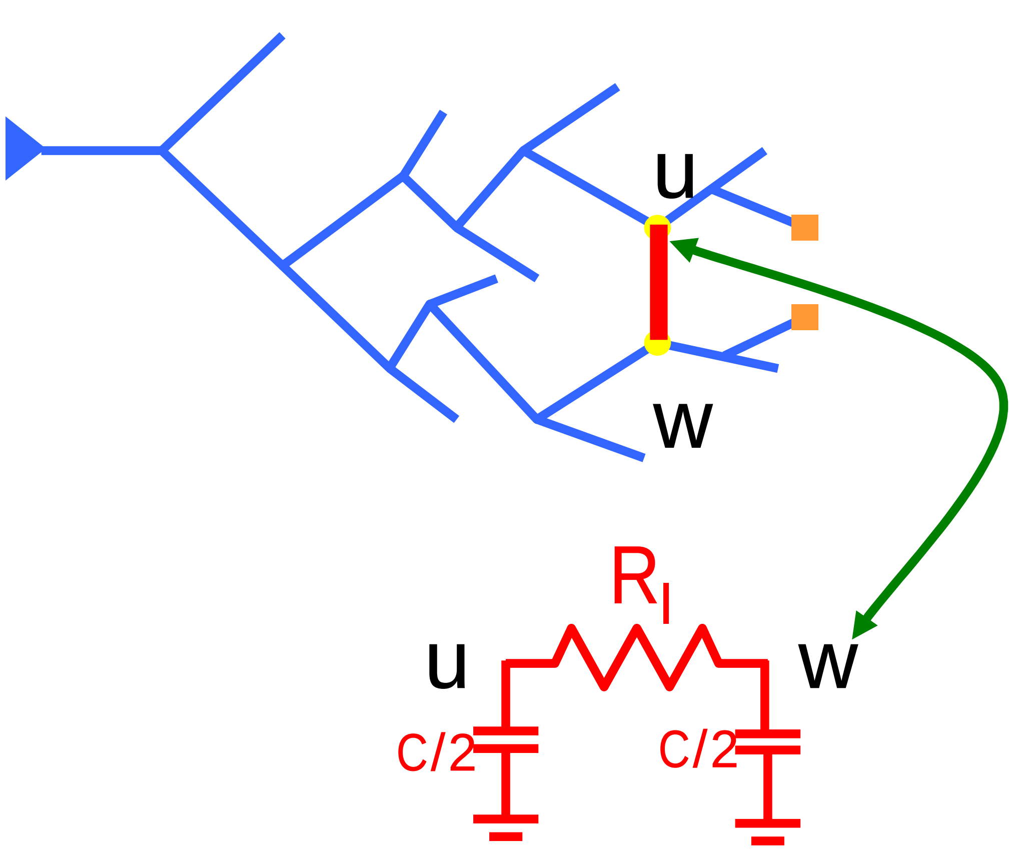
[Restle et. al, JSSC'01]

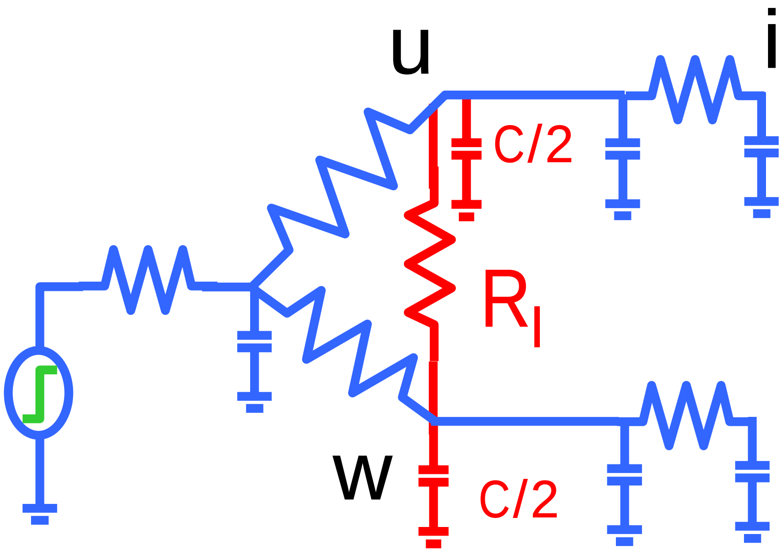
Applied in IBM microprocessor

- Very effective, huge wire

Non-tree: Link Perspective

- ♦ Non-tree = tree + links
- ♦ How to select link pairs is the key problem
- ♦ Link = link_capacitors + link_resistor





[Rajaram et al, DAC’04]

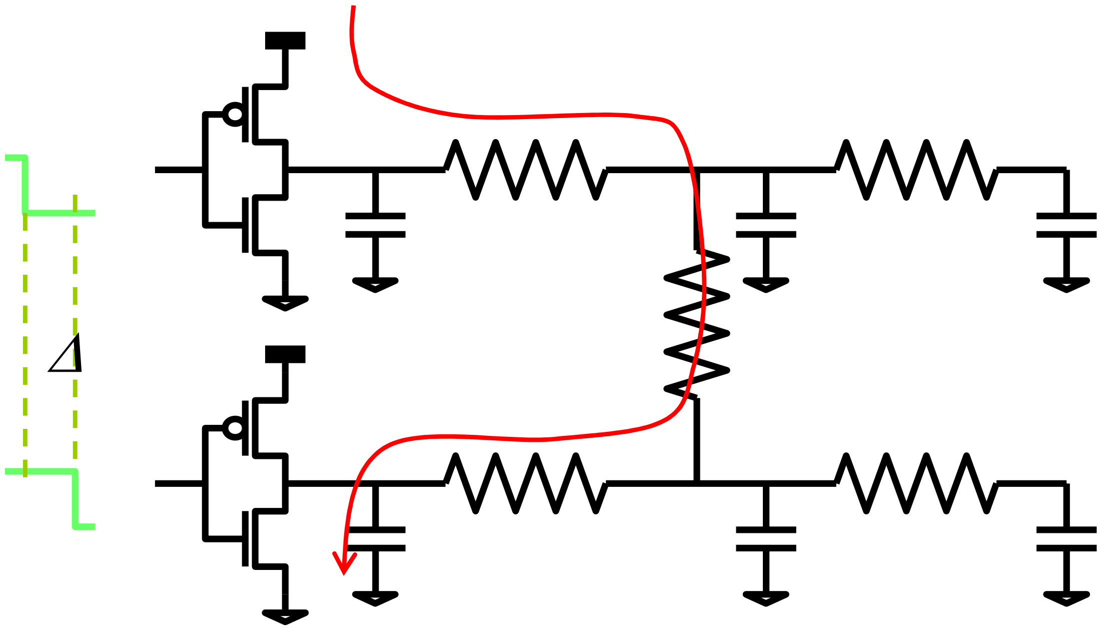
Guidelines for Link Insertion

[Rajaram et al, DAC'04]

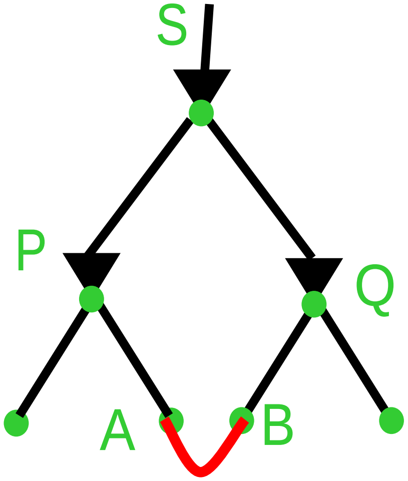
- Select nodes physically close to each other
- Select nodes which are hierarchically far apart
- Select nodes with equal nominal delay
- Select nodes closer to leaf nodes

Challenges for Buffered CTS

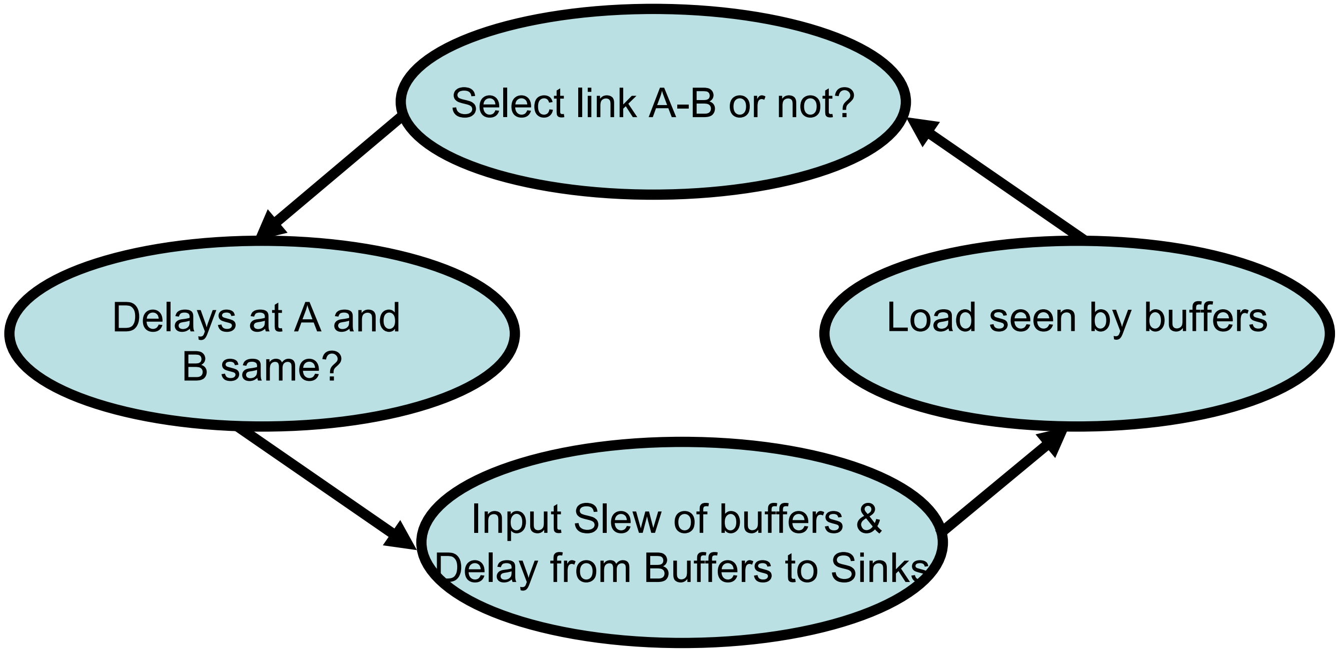
- Link insertion may cause multi-driver nets
  - Short circuit avoidance: $\Delta_{max} < Delay_{min}$ [Venkataraman+ ICCAD'05]
- Link insertion must have high delay accuracy cf. SPICE
  - Elmore delay not good fidelity cf. SPICE for buffered clock trees [Wang et. al, ISPD04]



Challenges for Buffered CTS



A Chicken-Egg problem!



Select link A-B or not?

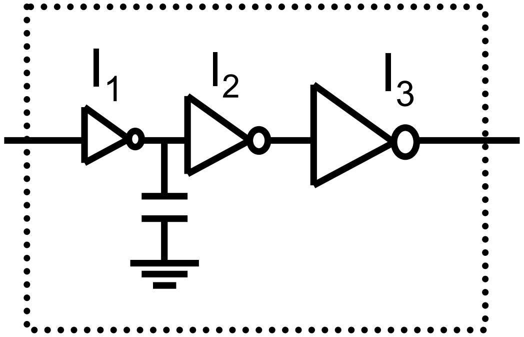
Delays at A and B same?

Input Slew of buffers & Delay from Buffers to Sinks

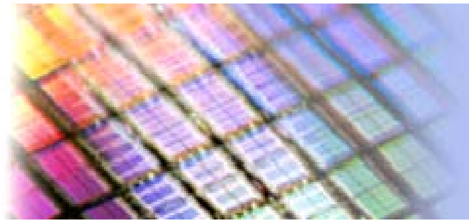
Load seen by buffers

Venkataraman+ ICCAD'05

- Addressed the problem of link insertion in buffered clock tree by
  - Using special tunable buffers to break the chicken-egg problem described before
  - Using SPICE to do the node tuning



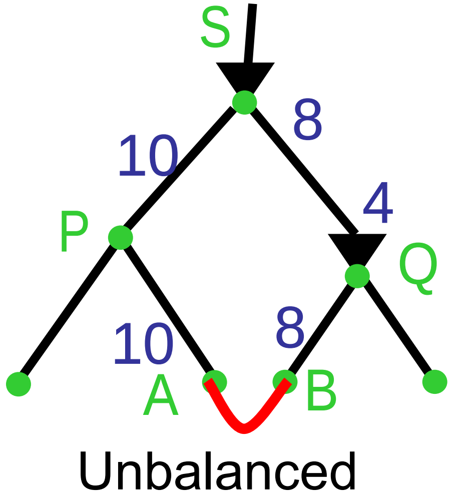
- Drawbacks:
  - Tunable buffers – not generally available
  - Will consume extra power/area due to extra capacitances in tunable buffers
  - Slow on very large clock trees due to use of SPICE



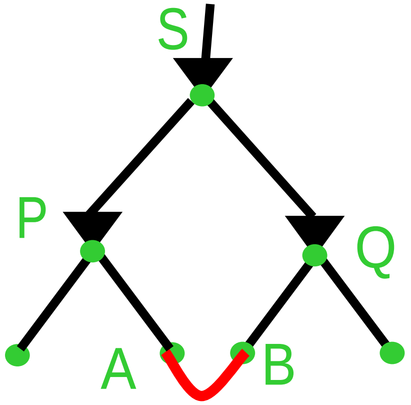
Our Contributions

- Link-insertion friendly balanced Clock Tree Synthesis algorithm
  - A new merging scheme for bottom-up CTS
    - guarantees balanced buffered clock tree while trying to minimize wirelength
  - Uses an Elmore like, but more accurate iterative delay calculator used by IBM [Puri et. al. GLVLSI’02] to break the chicken-egg dilemma
- Uses regular buffers instead of the tunable buffers of Venkataraman et. al ICCAD’05
  - Can be applied on any general design
  - No unwanted increase in capacitance/power

Why Balanced Clock Tree?



Unbalanced

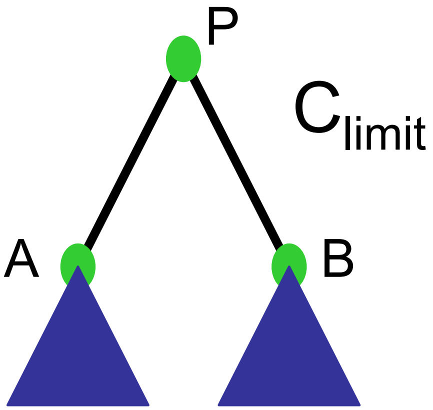


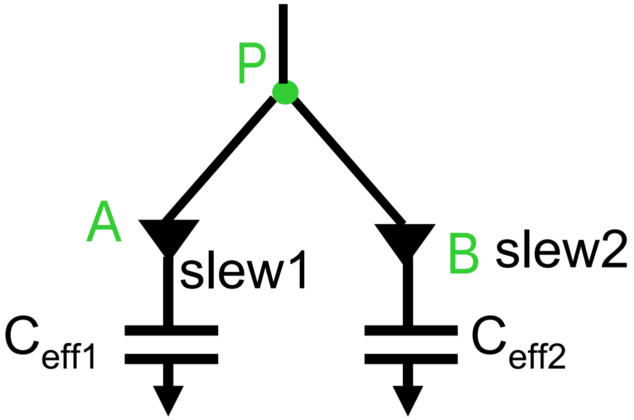
Balanced

- Current CTS Algorithms mostly focus on skews at nominal delay values
- Due to variation effects, delays and skews vary
  - Interconnect and Buffers have different variation patterns
- Having a balanced clock tree is likely to minimize the variational effects
- Balanced Clock Tree will reduce the possibility of short-circuit currents caused by link insertion

Balanced CTS Algorithm: Main Features

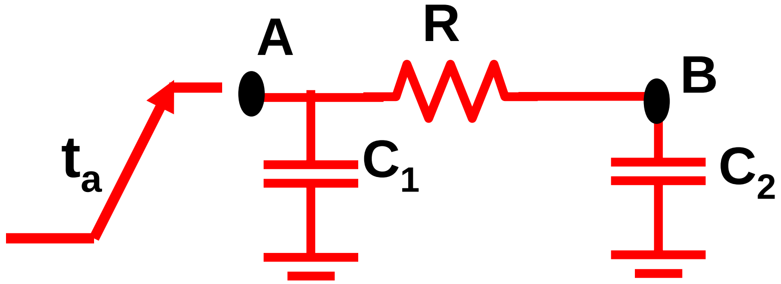
- Sub-trees A & B are merged only when the effective cap after merging is less than the cap limit C_{limit}
- Buffers are inserted at the root of all sub-trees if no merging is possible without violating the cap limit.
- The required slew information is propagated in a bottom-up manner for accurate delay calculation
  - Need accurate slew and C_{eff} information at buffer output





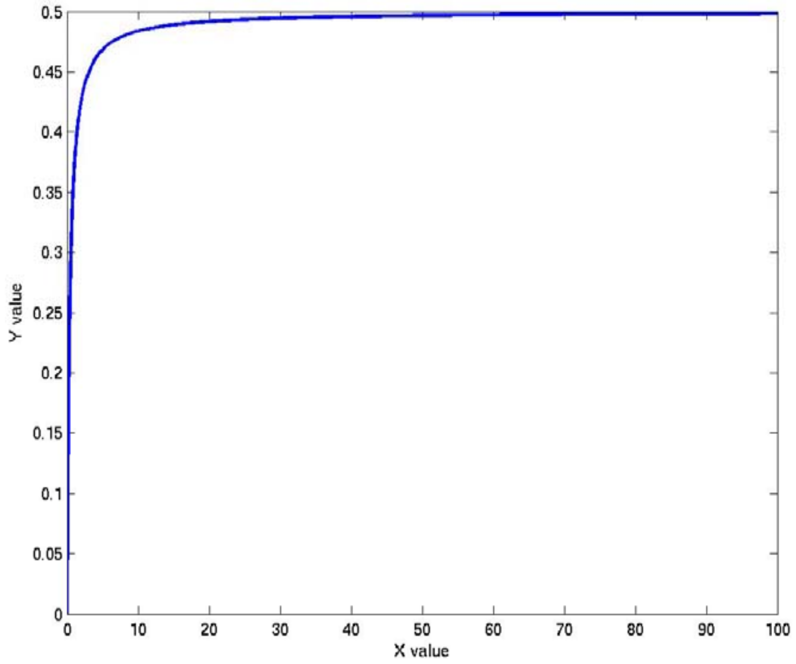
Backward Slew Propagation

- Based on Puri et. al. GLVLSI’02: given an input transition time t_a at node A, the slew at node B is given as:



$$t_b = \frac{t_a}{1 - x(1 - e^{-\frac{1}{x}})} \quad \text{where } x = \frac{R * C_2}{t_a}$$

$$\text{Let } \; y = \frac{R * C_2}{t_b}, \text{ then } \; y = x(1 - x(1 - e^{-\frac{1}{x}}))$$



- Value of y bounded by 0.5 for all x
- 1-1 correspondence (x, y)
- Given a t_b target, required t_a can be obtained

Pick Sub-trees to be Merged

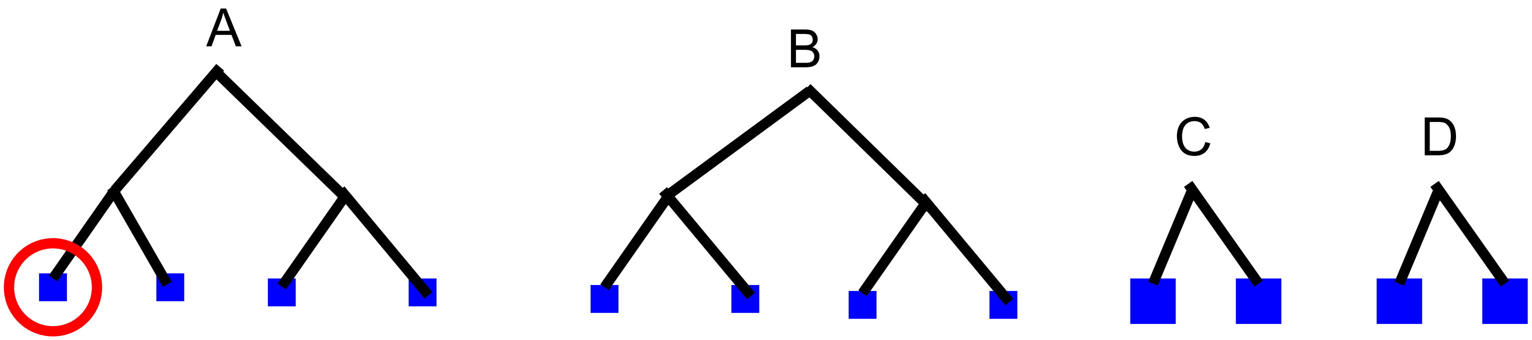
- Given *N* sub-trees to merge in list U:
  1. Pick the sub-tree with minimum root-sink delay - Ti
  2. Of all available sub-trees, pick Tj such that MergingCost(Ti, Tj) is minimized.
  3. Merge Ti, Tj to get Tk. Remove Ti, Tj from list U. Add Tk to list U.

Step 1 for delay balance. Smaller sub-trees will be merged first.

Step 2 for MergingCost (e.g. wirelength) minimization

Balanced CTS Algorithm

Sub-trees A,B, C and D cannot be merged without violating Climit

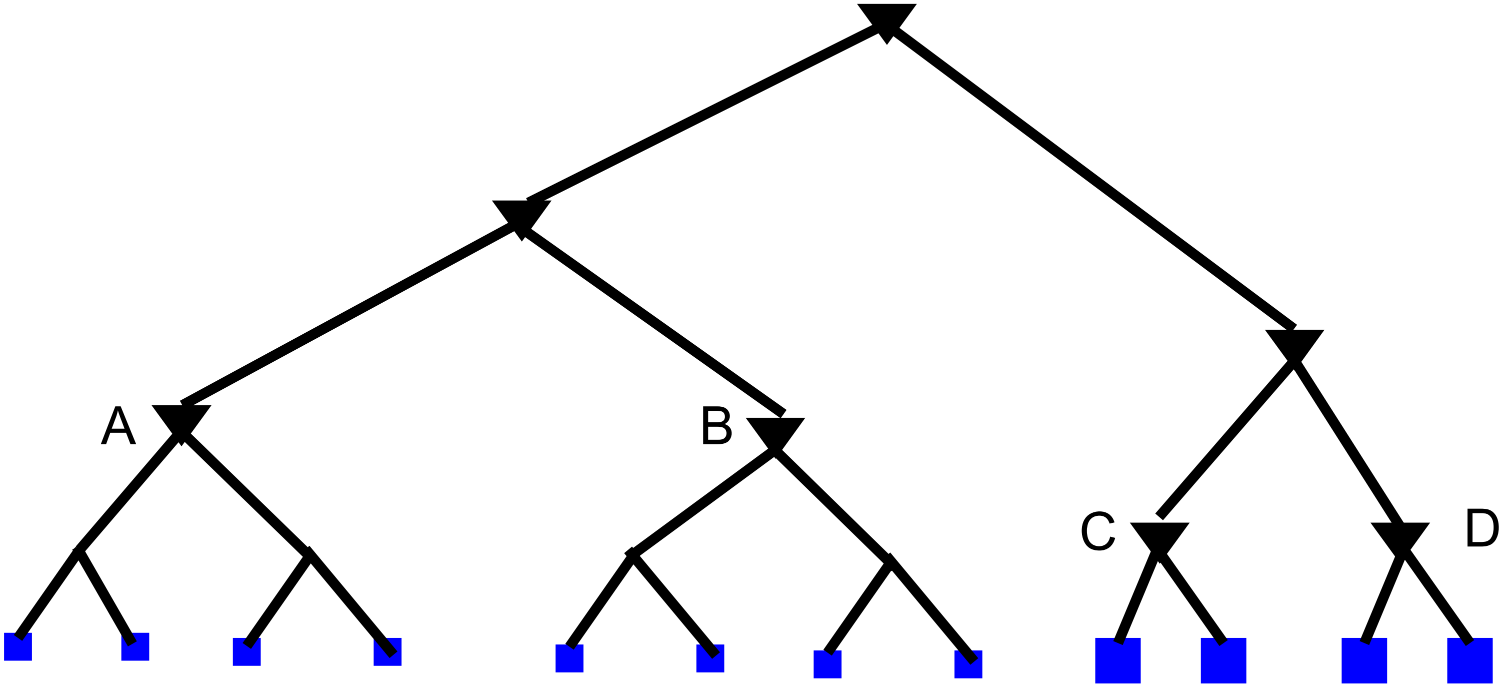


Pick the node with min delay. Initially, since all sinks have zero delay, pick the sink with min load cap.

Merge the picked node with another node such that merging cost is minimized Without violating Cap Limit

Repeat the process till no node pairs can be merged without cap limit violation

Balanced CTS Algorithm



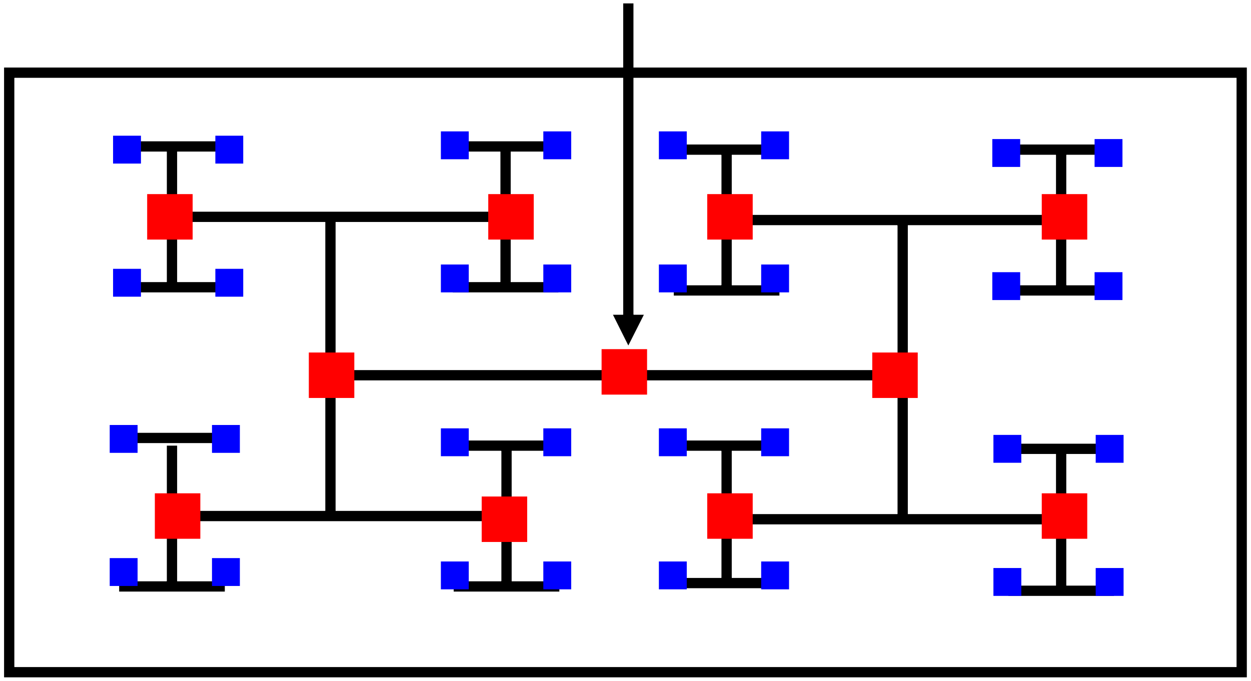
Buffer all the sub-trees at the same time. This guarantees balanced clock tree by construction. The load imbalance between buffers is also minimized.

Repeat the process to obtain the complete buffered clock tree

Overall Algorithm

1. Construct the balanced buffered clock tree (with accurate delay/slew model)
2. Select the link pairs for insertion using modified MST algorithm [Rajaram et. al. ISPD’05] that uses physical and delay proximity for link selection
3. Using link capacitance as extra sink load capacitance, and tune the clock tree with the same topology as in step 1
4. Add the link resistance to the selected node pairs

A Simple Example



🟦 Sinks

🟥 Buffers

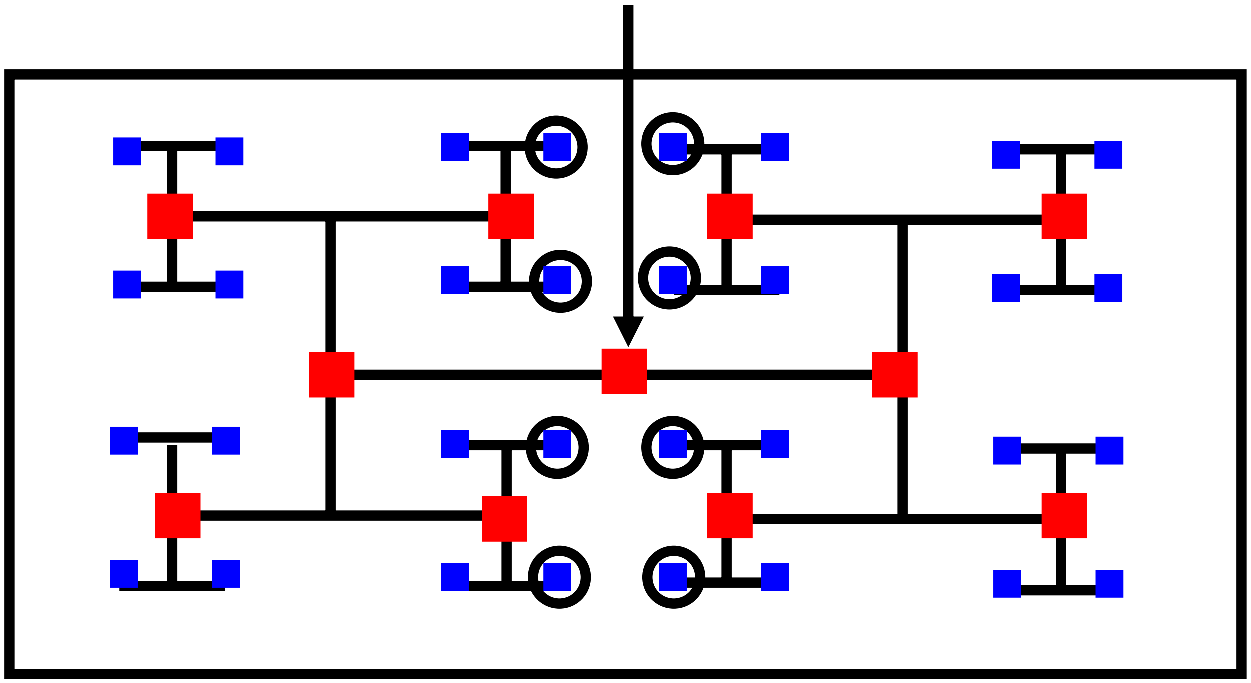
◙ Sinks selected for link insertion

● Sinks with added Link cap.

-∿∿- Link resistance

Construct a balanced clock tree

A Simple Example



🟦 Sinks

🟥 Buffers

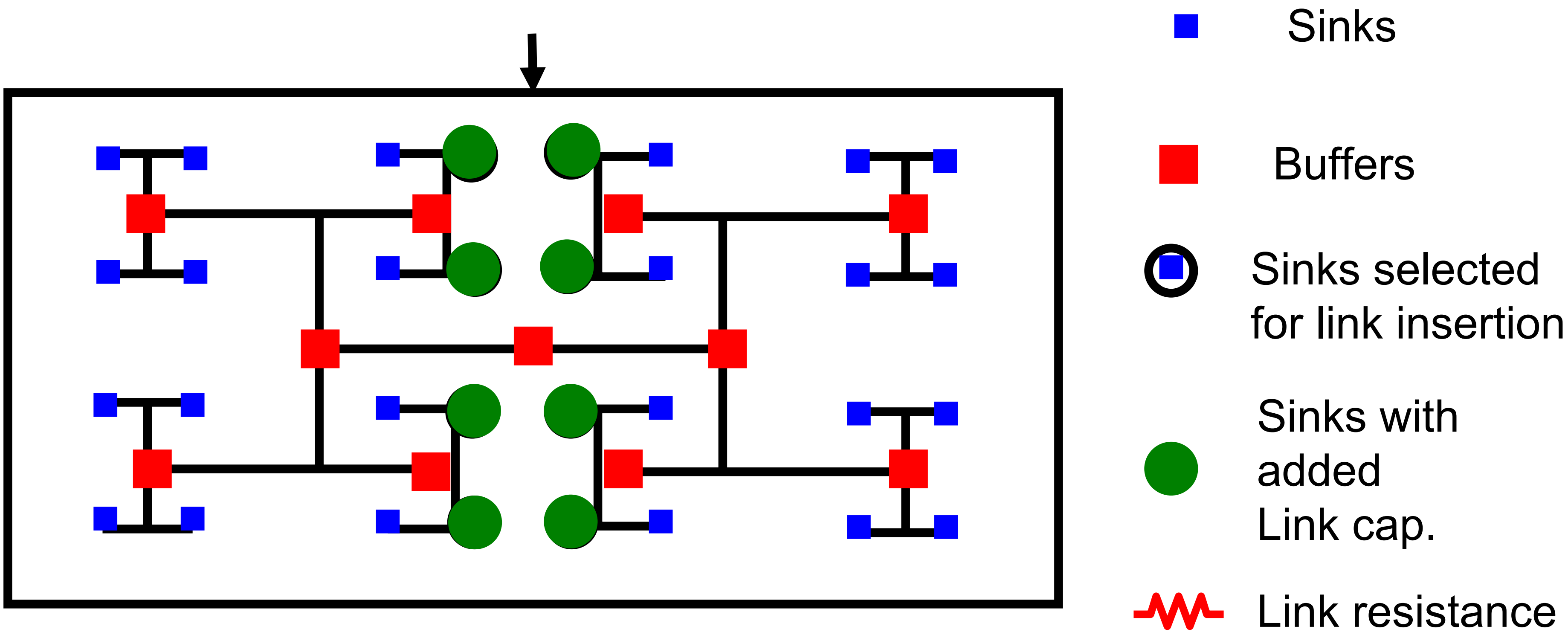
○ Sinks selected for link insertion

● Sinks with added Link cap.

-⩘- Link resistance

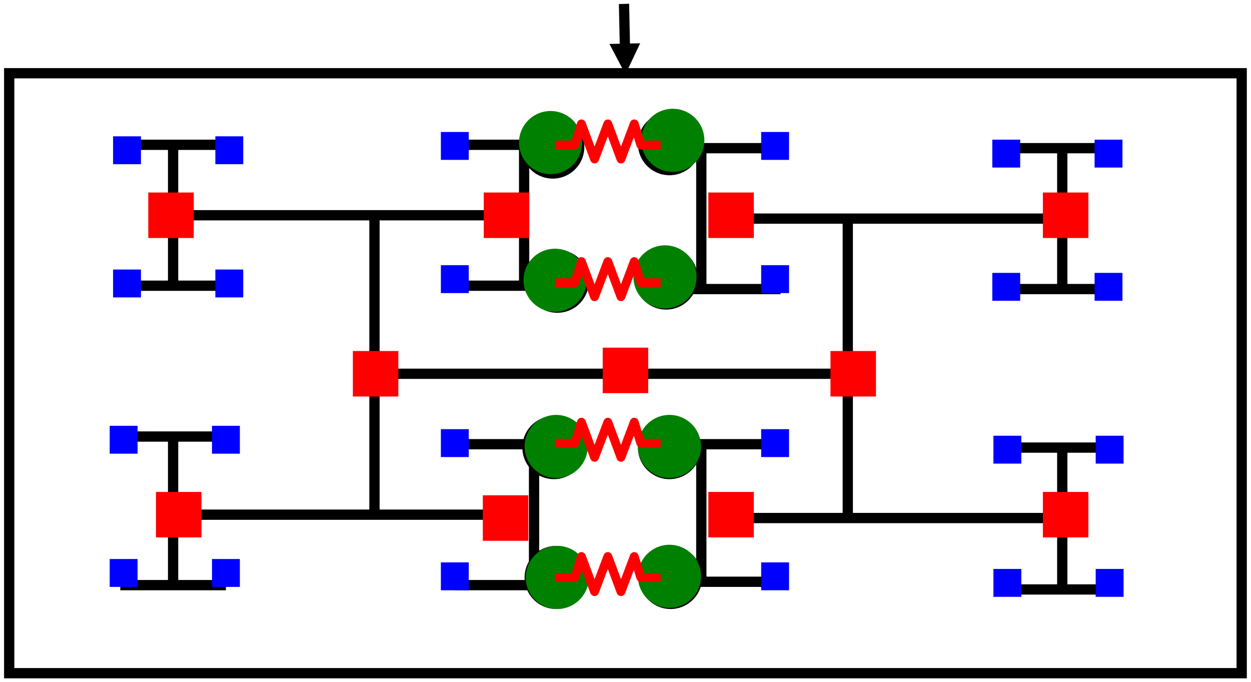
Select the link pairs for insertion using modified MST algorithm [Rajaram et. al. ISPD05] that uses physical and delay proximity for link selection

A Simple Example



Construct a new clock tree with the same topology as in step 1 using the balanced CTS algorithm

A Simple Example



🟦 Sinks

🟥 Buffers

◙ Sinks selected for link insertion

● Sinks with added Link cap.

-W- Link resistance

Add the link resistance to the selected node pairs

Experimental Setup

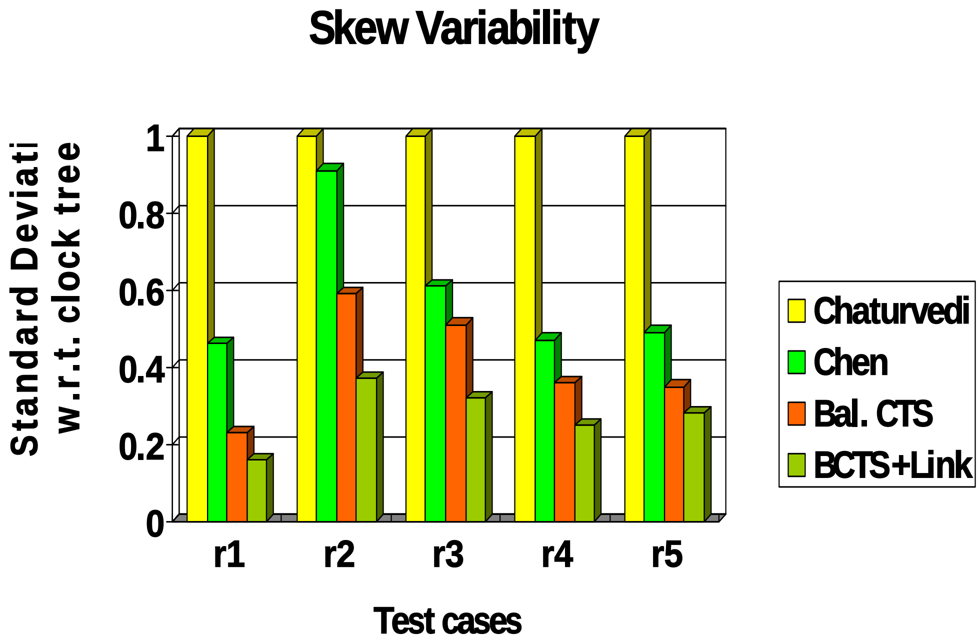
- Benchmarks: r1-r5 from Exact Zero Skew work [Tsay, ICCAD’91]
- Variations considered (σ = 5%)
  - Buffer L, Tox
  - Interconnect width
  - Load Capacitance
- Skew variability measure: Average magnitude of skew in SPICE with 500 Monte Carlo trials.

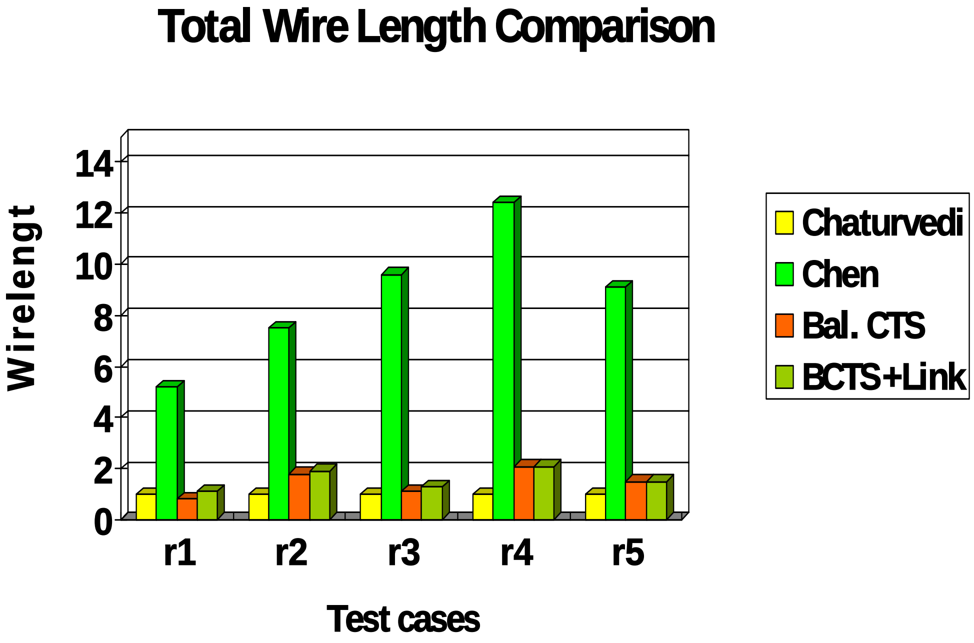
Benchmark	r1	R2	r3	r4	r5
No. of sinks	267	598	862	1903	3100

Experimental Setup

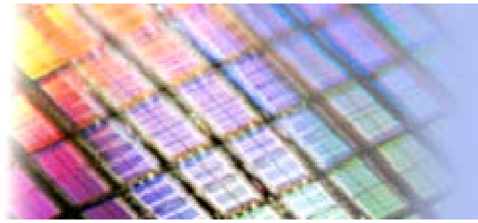
- Results compared to
  - Chen et. al, DATE’96 (balanced CTS)
    - Equalizes delay at each stage of clock tree by wire elongation
    - Excessive wire elongation results in excessive wire length
  - Chaturvedi et al, ISQED’04 (best wire-length for CTS)
    - Results in unbalanced clock tree
- Cannot compare with [Venkataraman+ ICCAD’05] directly
  - Special tunable buffers not available
  - Small benchmarks used in their work (running SPICE directly to construct/tune the clock network)

Experimental Results





- ♦ All results normalized w.r.t. Chaturvedi et. al
- ♦ The number of buffers used are similar
- ♦ All three algorithms were tuned to achieve the same slew rate requirements



Conclusions

- We have proposed a link insertion friendly balanced buffered CTS algorithm
- Ordinary buffers are used (instead of special tunable buffers)
- Our merging scheme achieves balanced clock tree without excessive cost of wire length
- Skew variation is significantly reduced
- Link insertion becomes more practical even for ASICs…