WBA Algorithm-

Wire sized Buffered A-Tree Algorithm

WBA:

- Constructs a Steiner tree while minimizing arrival time (timing slack) at source
- Performs Steiner Tree construction, buffer insertion and wire sizing simultaneously
- Elmore Delay model used for interconnects and RC Delay model for buffers:

$$D_{wire}(e_v) = r_{e_v}(c_{e_v}/2 + c(T_v))$$

$$D_{buff}(b, c_l) = d_b + r_b.c_l$$

Problems with independent approaches:

- Buffer Insertion, then Steiner tree-wiring delay routability not accurate.
- Steiner Tree, then buffer insertion and wire sizing-not necessarily a min. delay Steiner tree

Okamoto and Cong, "Buffered Steiner Tree construction with Wire Sizing"

Optimization Objectives

- Required arrival time at the root of a tree T_v:

$$q(T_v) = \min_{u \in sinks(T_v)} (q_u - delay(u,v))$$

(q_u -> required arrival time at sink u;
$sinks(T_v)$ -> set of sinks of tree T_v;
delay(u,v) -> delay from u to v)

- Total Capacitance of T_v :

$$c_{total}(T_v) = \sum_{e \in Tv} c_e + \sum_{u \in buffers(Tv)} c_u + \sum_{u \in sinks(Tv)} c_u$$

(buffers(T_v) -> set of buffers in T_v;
c_u -> loading capacitance of buffer or sink u)

Wire Sized buffered Steiner Tree Problem Formulation

- *KNOWN*: Source s_0, sinks s_1, s_2,…s_n of signal net S (given positions and required arrival time for s_i ($1 \le i \le n$)
- *TO FIND*: A Steiner Tree T_{s_0} that connects S and has wires sized and buffers inserted
- *OBJECTIVE*: To maximize $q(T_{s_0})$[Primary goal] while minimizing $c_{total}(T_{s_0})$[Secondary goal]

- *ASSUMPTIONS*:

  Only one type of buffer is considered for insertion and signal polarity is neglected

Quick Review of the A-Tree and Buffer Insertion&Wire Sizing Algorithms used in WBA

- **A-Tree** Algorithm:

  Definition: If every path of a rectilinear Steiner tree connecting source s_0 and any node p is the shortest path -> A-Tree.

  Only *heuristic* A-Tree moves used.

  'ROOT' : has a set of roots of current sub-trees to be merged into one Steiner tree. The algo iteratively merges a pair of roots at a point far away from the src, till ⊠ROOT⊠ =1.(All sinks assumed to be in the 1st quadrant)

  Procedure Heuristic_Atree()

1. Root<- {s_i⊠0≤ i≤ n}
2. While ⊠ROOT⊠ >1 do
3. Find u,w□ROOT so sum(min(u_x,w_x)+min(u_y,w_y) is a max
4. ROOT <- Root + {v}-{u}-{w} [v:node with co-ord. (min(u_x,w_x),min(u_y,w_y))]
5. Merge T_u&T_w-> T_v; add edges from v->u,w respectively.

   End while

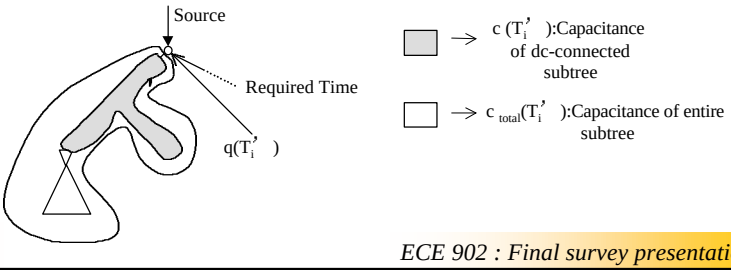
   End procedure

Buffer Insertion and Wire Sizing Algorithm

- Chooses the buffering position to maximize the required arrival time at source while integrating wire sizing and total capacitance
- Based on a dynamic Programming technique with 2 phases:

1. Bottom up: Finds all possible potions for node('candidate points') in a bottom up manner, choosing the best option finally.
2. Traces back computations from 1 and finds buffer position and wire width for each edge.

Assumptions: Topology of routing/Steiner tree given; wire widths and possible positions for insertion change.



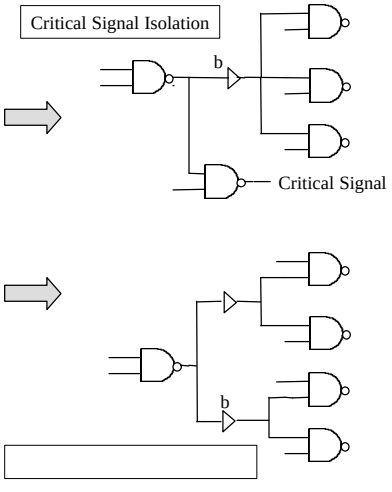
WBA -Wire sized Buffered A-Tree algorithm

Combines the two algorithms for simultaneous tree construction, buffer insertion and wire sizing

- For fanout optimization:

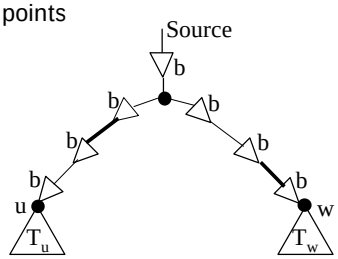
1. 'Critical path isolation' :
   -one/several sinks are timing critical
   -root gate drives critical sinks and s smaller load due to buffered non-critical paths
2. 'Balanced Load Decomposition' :
   -required times at the sinks are within a small range
   -load at output of the root gate is reduced

- Used in choosing two sub-trees(T_u,T_w) to be merged to T_v in the A-tree algorithm



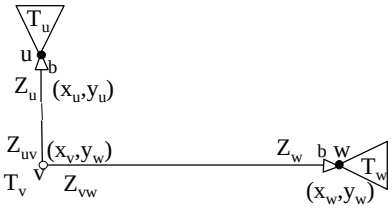
Fanout Optimization(contd)

- ’
  location(v) and current required time at each sub tree’ s root(v,w), and the best selected for min wire length
- *Iterations:*
  Select u,w
  Merge T_u,T_w->T_v,compute set of options at v by bottom_up(T_v)
- Edges with length more than a threshold are divided recursively to insert buffers in the middle of a wire; for wire width changes, the same points are used as candidate points



Evaluation of Sub trees to be merged

The computation of options(which is the 1st stage in both A-tree and buffer insertion and wire sizing) is done simultaneously:



- Consider Z_u,Z_w-->set of options at u,w(already computed)
- l_{e_u},l_{e_w}=0 (parent nodes of current sub tree’ s roots not yet found)
- Z_u <-- Z_v’ ; for the merge, find Z_{vu}, Z_{vw} (assumption:parents of u,w->v)
- Idea of sub tree merge: When the min value of required times of two sub trees is max along current possible merging pairs, critical path isolation and balanced decomposition are achieved

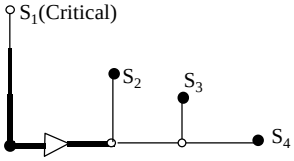
Basic definitions in WBA

- Q_{uw}=min(max q_z, max q_z)
  $z \in Z_{vu}$ $z \in Z_{vw}$
- $Q_{max}(ROOT)$= max Q_{uw}
  $u,w \in ROOT$
  (Merge so Q_{uw} is large to maximize time at root)
- D_{uw}=min(u_z,w_x)+min(u_y,w_y)
  (For 1st quadrant, s_0 at origin)
- $D_{max}(ROOT)$= max D_{uw}
  $u,w \in ROOT$
  The heuristic A-Tree algo always chooses the pairs of sub trees Tu and Tw with
  D_{uw} =$D_{max}(ROOT)$ (effective for wire length minimization)

- **Merging Cost :** mcost(u,w,ROOT)=
  $\alpha(Q_{uw}/Q_{max}(ROOT))+$
  $(1-\alpha)(D_{uw}/D_{max}(ROOT))$
  where α ->fixed constant($0 \le \alpha \le 1$)
  - For subtree merge, choose T_u,T_w to max mcost
- Why mcost??
  Since the required time minimization with buffer insertion and wire length minimization are achieved simultaneously
  1. The 1st term contributes to critical path isolation and balanced load decomposition
  2. The 2nd term minimizes the wire length as the original A-tree
- When critical sinks are involved, they are isolated

Example of a WBA Tree

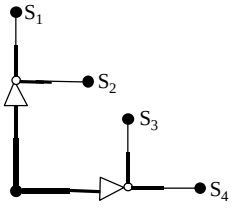
- Tree with Critical sink isolation



S_1 is critical. So, isolate since the merge results in a smaller Q

The layout/geometric information is taken into consideration due to the D term in the merging cost

- Tree with balanced load decomposition



Required times at sinks s_1, s_2,s_3,s_4 are within a small range

Load is balanced, since Q of the merge for the sinks are within a small range

2 Phases of the WBA Algorithm

- Bottom_up tree construction+option computation

Procedure WBA_tree_bottomup()

1. ROOT<--{s_i ⊠ 0 ✂ i ✂ n}
2. Foreach v □ Root do
3. Bottom_up(v) (v for each sink)
   endfor
4. While ⊠ROOT⊠ >1 do
5. Find u,w □ ROOT with max $_{u,w}$ □ROOT mcost(u,w,ROOT);
6. ROOT <- Root + {v}-{u}-{w} [v:node with co-ord. (min(u_x,w_x),min(u_y,w_y))]
7. Merge T_u&T_w-> T_v; add edges from v->u,w respectively.
8. bottom_up(Tv);

   /*Z_v⊠l_{e_v} =0 is computed with pruning, Z_u<--Z_{vu}, Z_w<--Z_{vw}*/

   end while;

end procedure

2 Phases of the WBA Algorithm (contd)

- Top_down phase:

  Buffer insertion and wire width assignment takes place in this stage

- The option with the maximum required time and minimum total capacitance at the root is chosen
- Computations are traced back to the first phase that led to this option
- During this backtrace, the buffer positions are wire width for each of the segments is determined

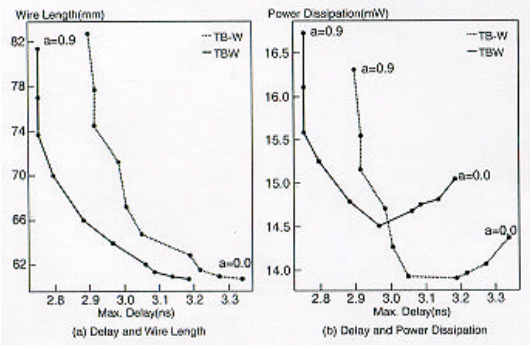
Lillis,Cheng and Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model"

Experimental Results Obtained

- The WBA was implemented under C/UNIX environment, with HSPICE used to verify results
- Technical parameters used were based on: MCNC 0.5u CMOS process model
- One type of buffer and set of wire widths(W,2W,3W) were used
- Source driven by an ideal voltage source; input signal a square wave of 25MHz

Experimental Results Obtained (contd)

	T+B+W			T+BW			TB+W			TBW		
n	d(ns)	p(mW)	w(mm)	d(ns)	p(mW)	w(mm)	d(ns)	p(mW)	w(mm)	d(ns)	p(mW)	w(mm)
10	2.28	4.86	25.3	2.04	5.13	35.4	2.30	4.83	27.0	1.88	5.21	26.9
25	2.86	8.96	42.4	2.64	9.45	63.4	2.70	9.16	47.5	2.51	9.68	47.1
80	3.54	14.37	80.7	3.19	15.05	99.7	2.98	14.71	73.2	2.80	15.35	70.0



This approach is found to outperform other existing approaches by upto 16% in terms of maximum delay in trees.

NonHanan Routing-A brief Overview

- A performance driven routing formulation whose objective is to meet a specified delay constraint at each sink
- RLC circuits
- NonHanan points considered for minmax problem and to achieve a specified delay at each sink
- Elmore delays; SERT method(greedy Steiner Tree)
- **MVERT**(Maximum delay Violation Elmore Routing Tree) Algorithm
- 2 Phases:

  Initial tree construction(minimize delay)

  Cost-improvement

Hou,Hu and Sapatnekar, "NonHanan Routing"