Linking Layout to Logic Synthesis: A Unification-Based Approach

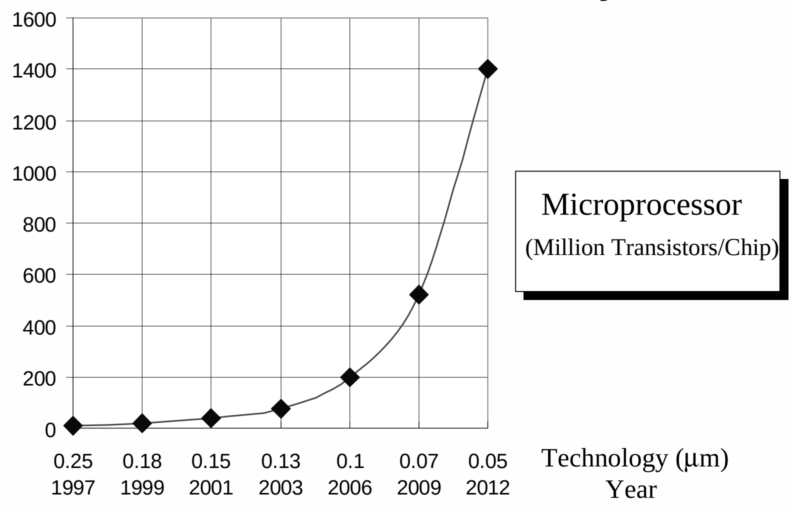
Massoud Pedram
Department of EE-Systems
University of Southern California
Los Angeles, CA

February 1998

Outline

- Introduction
- Technology and Design Trends
- EDA Flows
- Requirements for DSM Design Tools
- Simultaneous Mapping and Placement
- Conclusion

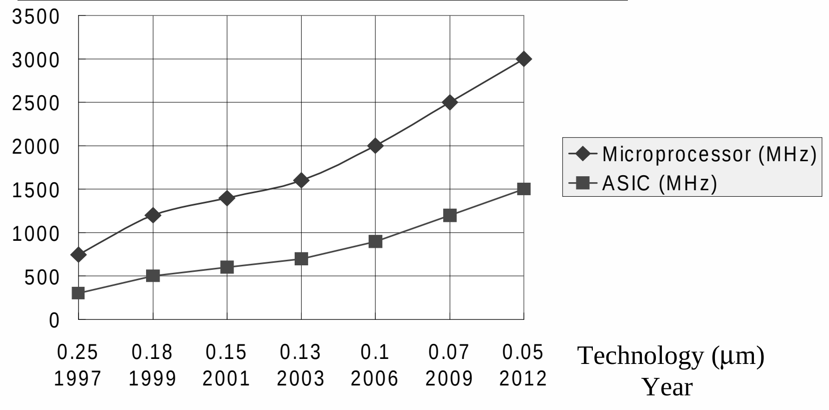
Trends: Circuit Density



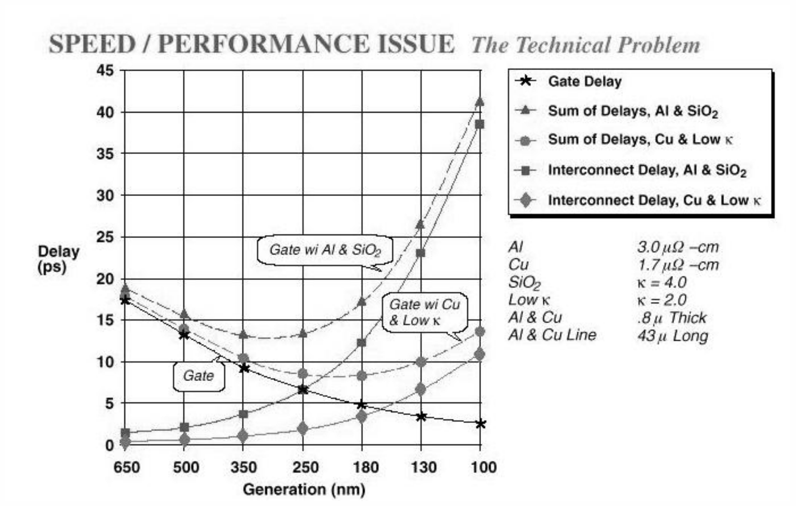
Extracted from NTRS, 1997

Internal Clock Frequency

On-chip, across-chip clock, high-performance

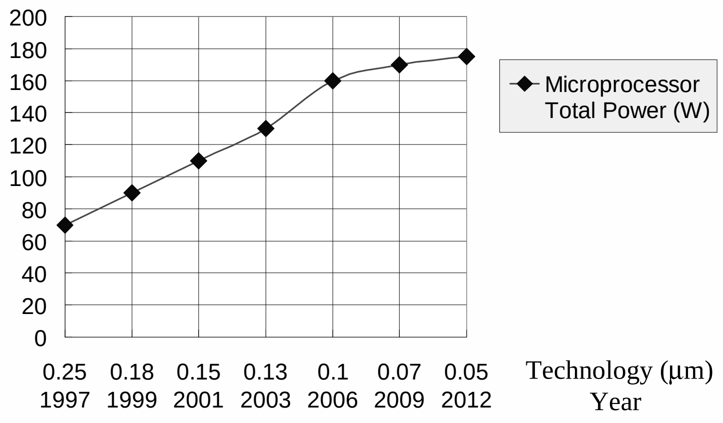


Interconnect Dominance



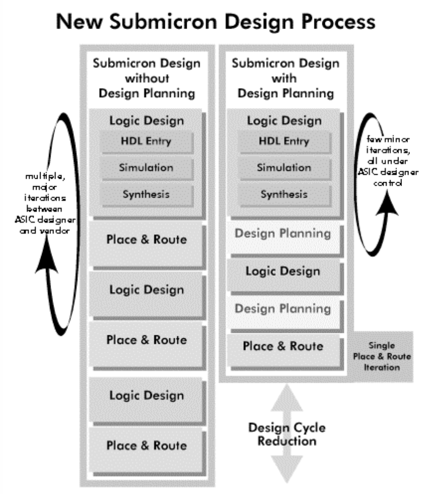
Extracted from NTRS, 1997

CPU Power Dissipation



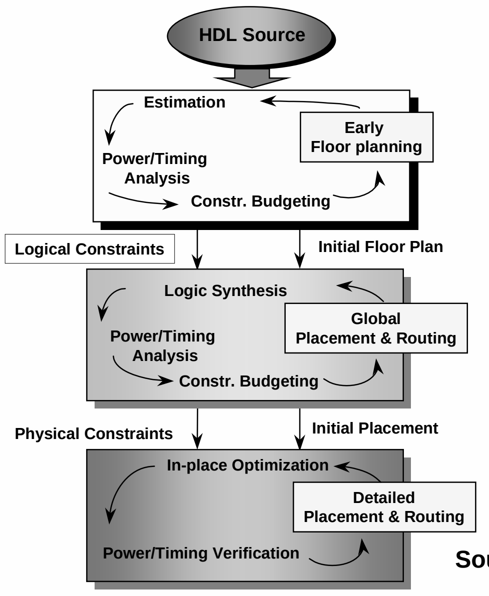
*Extracted from NTRS, 1997 (*Assuming 2x minimum width and spacing)*

Design Planning Flow


New Submicron Design Process
Submicron Design without Design Planning
Logic Design
HDL Entry
Simulation
Synthesis
Place & Route
Logic Design
Place & Route
Logic Design
Place & Route
multiple, major iterations between ASIC designer and vendor
Submicron Design with Design Planning
Logic Design
HDL Entry
Simulation
Synthesis
Design Planning
Logic Design
Design Planning
Place & Route
few minor iterations, all under ASIC designer control
Single Place & Route Iteration
Design Cycle Reduction

Source: Cadence

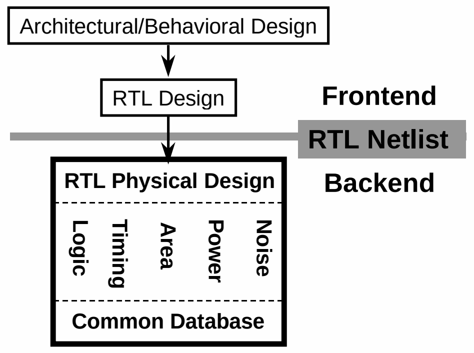
Synthesis-Centric Flow



Source: Synopsys

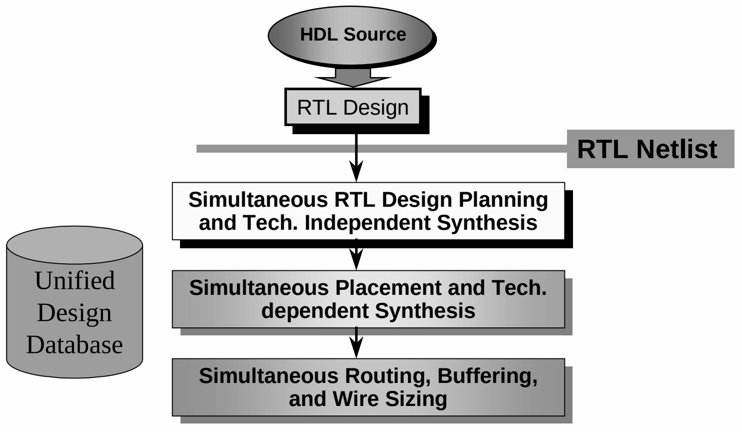
Physical Design-Centric Flow

Interconnect Delay Dominant

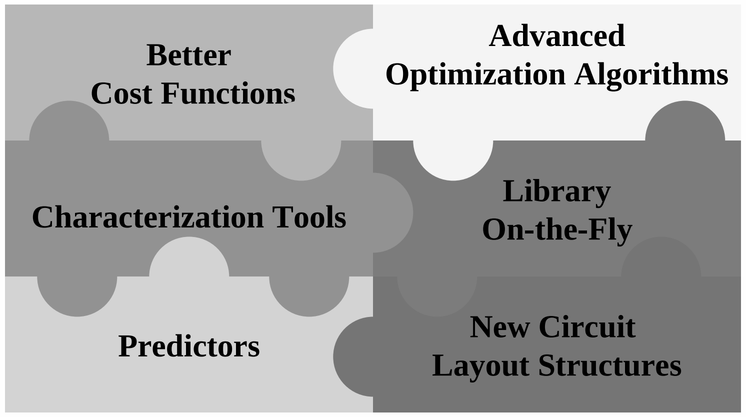


Source: Avant!

Unification-Driven Flow

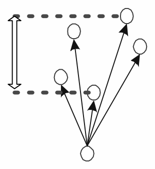

HDL Source
RTL Design
RTL Netlist
Simultaneous RTL Design Planning and Tech. Independent Synthesis
Simultaneous Placement and Tech. dependent Synthesis
Simultaneous Routing, Buffering, and Wire Sizing
Unified Design Database

Design Flow Enablers

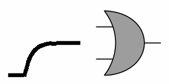


Better Cost Functions

Minimize interconnect cost rather than literal savings during kernel extraction



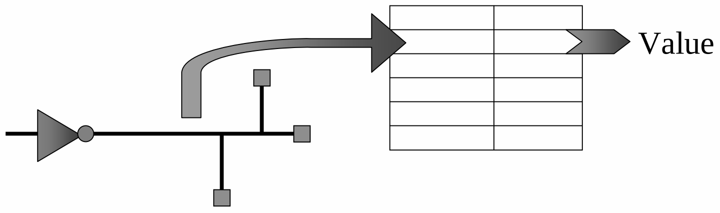
Account for signal transition times during technology mapping and placement



Use a high order moment matching model for interconnect delay calculation during routing

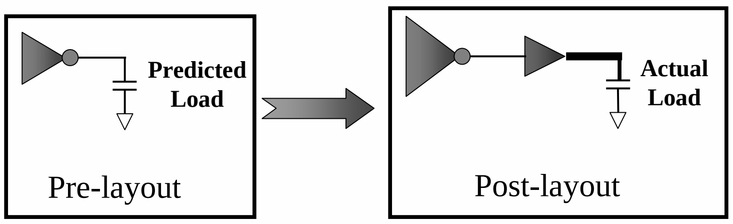
Passive Predictors

- Gate and interconnect area estimates
- Statistical wire load models
- Zero-delay power estimates



Active Predictors

- A constant delay model
  - Forward-annotate predictions by using gate sizing, buffer insertion, wire sizing, etc.



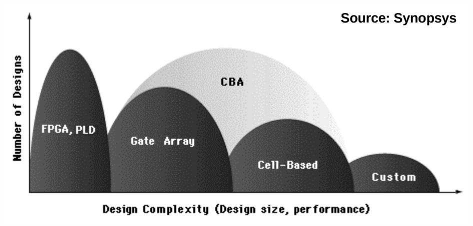
Advanced Optimization Algorithms

- Integration of floorplanning and timing-driven logic partitioning / restructuring
- Concurrent technology mapping and gate placement
- Simultaneous critical-sink Steiner tree construction and buffer insertion / sizing
- Concurrent gate placement and sizing for high performance

New Circuit Structures

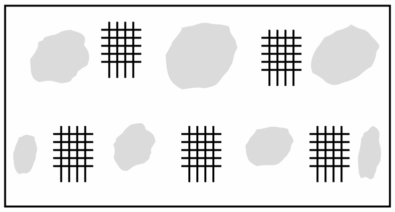
Gate-Centric Layout Styles:

- Cell-Based Arrays



Interconnect-Centric Layout Styles:

- Plan routing resources
- Place mapped gates between routed areas



Library Design

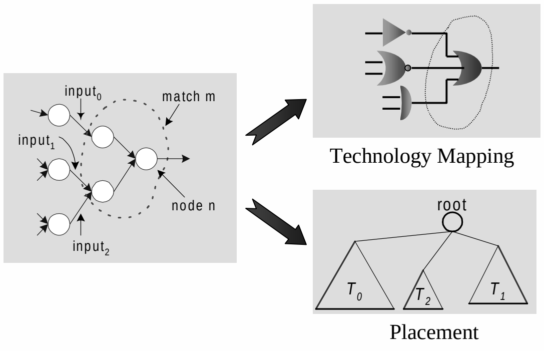
- Low power versus high performance
- Static versus dynamic
- CMOS complementary versus pseudo-NMOS
- Library-based versus on-the-fly-synthesized
- Characterization and modeling issues

Putting It Together

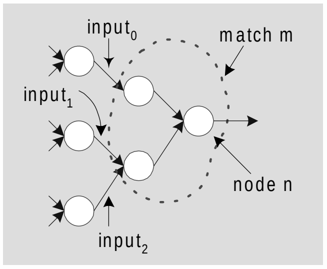
- Front-end optimization tools must be able to cope with the increasing complexity of DSM circuits
- Back-end analysis tools should handle complicated second-order effects in DSM circuits
- Must have
  - interconnect-optimized process technologies
  - new circuit layout structures
  - interconnect-driven design flows, algorithms and tools
  - signal integrity modeling and characterization tools
  - ability to handle multiple constraints at all levels of abstraction
  - industry standards

SiMPA: Problem Formulation

Given a tree network *T* and a library *L*, find a simultaneous technology mapping and linear placement implementation such that some cost functions are optimized



Technology Mapping



MinAreaTechMap (*N*,*L*) [Keutzer, 87]
INPUT: a tree network *N*, a library *L*
OUTPUT: a mapped network

```
1. Decompose N
2. Perform a reverse depth-first-search from
   primary inputs to the primary output
3. For each node in the reversed DFS order
4.     For every match m of n
5.         gate_area = sum of accumulated gate_area of all inputs
                        of this match + gate area of this match
6.     Store the best gate_area in n, the match m and the inputs of m
7. Get the best area solution from the primary output, and recursively build the
   mapping solution for all inputs which lead to this best solution
```

☞ **KA finds the minimum gate area mapping for *N***

Linear Placement

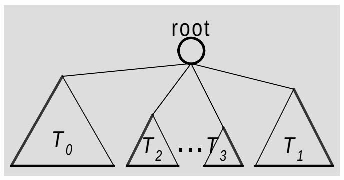
ApproxMincutPlace (root, t_0, t_1, t_2 … t_k) [Lengauer, 82]

INPUT: A root node and a list of placed subtrees

OUTPUT: A placed tree consisting of the root and all the subtrees

1. *Sort the subtrees in non-increasing order based on their maximum cut width (tie breaker gives to the trees that have balanced cut width), rename them as T_0, T_1, T_2, … T_k*

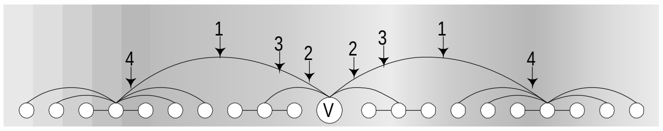
2. *Return a placement P as: T_0 T_2 … root T_3 T_1, such that for each T_i, the side with lower cut width is facing the root.*



☞ **LA finds a placement that is at most 2X away from the optimal solution**

Cut Cost Function

CCF extends the concept of cut width

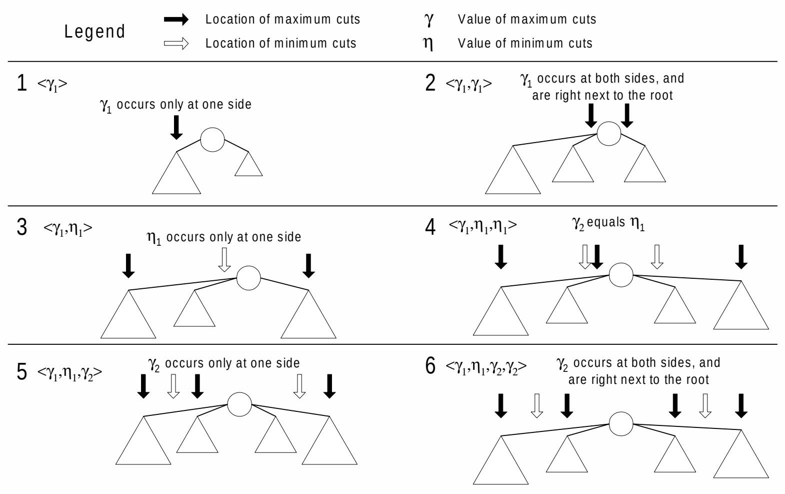


Example for calculating *CCF(T)*: maxCut value minCut value *CCF(T)*=<4,1,3,2,2>

$$|CCF(T)| \le num(T)+1$$

$$|CCF(T)| \le cut(T)+2$$

CCF Signatures


Legend
Location of maximum cuts
Location of minimum cuts
γ Value of maximum cuts
η Value of minimum cuts
1 <γ1>
γ1 occurs only at one side
2 <γ1,γ1>
γ1 occurs at both sides, and are right next to the root
3 <γ1,η1>
η1 occurs only at one side
4 <γ1,η1,η1>
γ2 equals η1
5 <γ1,η1,γ2>
γ2 occurs only at one side
6 <γ1,η1,γ2,γ2>
γ2 occurs at both sides, and are right next to the root

Yannakakis' Algorithm

<u>ExactMincutPlace</u> (root, t_0, t_1, t_2 … t_k) [Yannakakis, 85]

INPUT: A root node and a list of placed subtrees

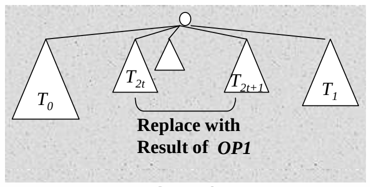
OUTPUT: A placed tree consisting of the root and all the subtrees

1*. Sort the subtrees in non-increasing order based on their CCF's, rename them as T_0, T_1, T_2, … T_k. Form a placement P similar to Lengauer's Algorithm*

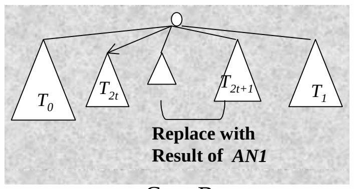
2*. Return P unless one of the following case is true:*

Case A*: If the maximum cut occurs only once (must be on the left side) and is inside the tree T_{2t}, then return the placement as below.*

Case B*: If the maximum cut occurs on both sides, then choose the tree on the right side with the largest index, say T_{2t+1}, then return the placement as below.*



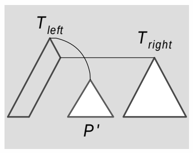
Case A

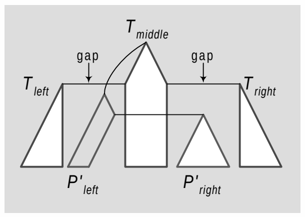


Case B

YA (cont)

OP1 and *AN1* break the first tree and place the remaining trees in its gap(s) as shown below

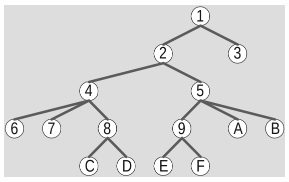




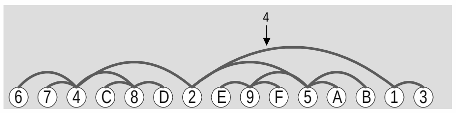
☞ **YA finds the optimal linear placement for *N***

☞ **The *CCF* of the placement is monotonic in the *CCF* of its subtrees**

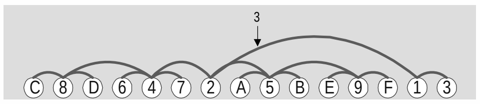
An Example



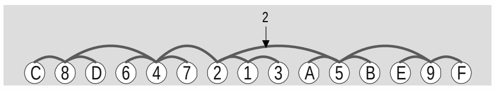
An example with 15 nodes



With a simple algorithm, the cut width is 4



With Lengauer’s algorithm, the cut width is 3



With Yannakakis’ algorithm, the cut width is 2

Area Cost Functions

Area Cost Function: $\boxed{A = W \cdot (h + \beta \cdot c)}$

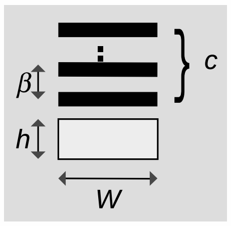
where:

A is the total gate area

$W = \sum_{cell_i} width(cell_i)$

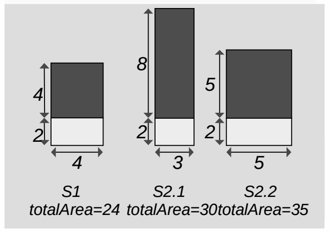
β is the minimum distance between the center of two adjacent wires
β = (minWireWidth+minWireSpacing)

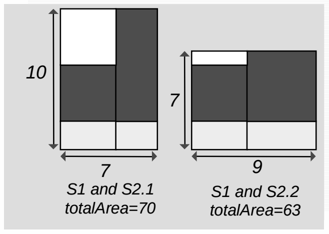
h is the cell height
c is the maximum cutwidth



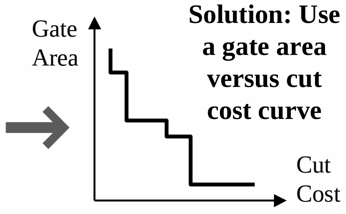
Gate Area versus Cut Cost Curve

A counter example shows why we cannot simply use area cost function for DP:

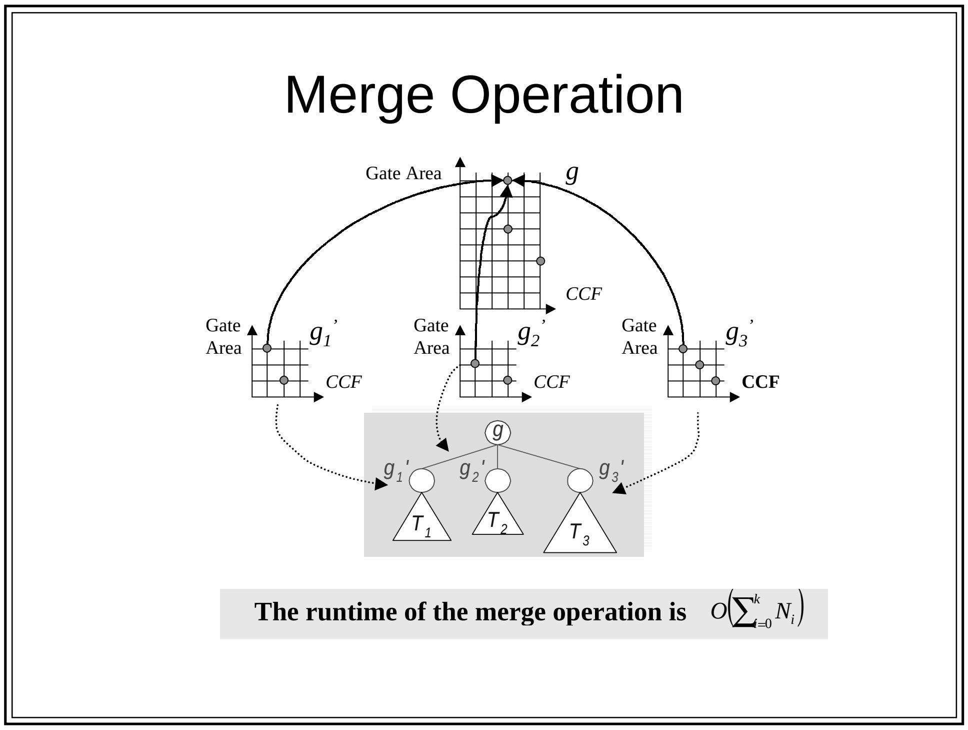




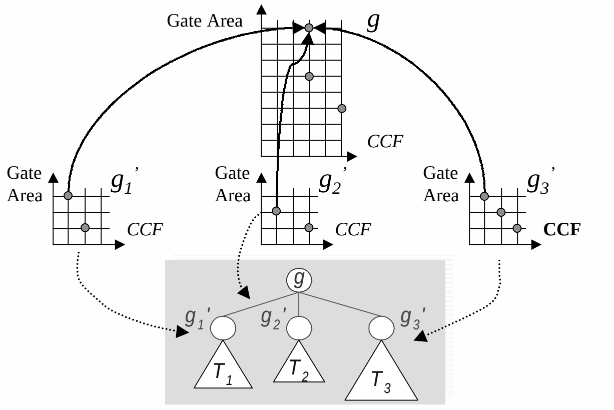
☞ **Direct combination of KA and YA does not produce the optimal solution**



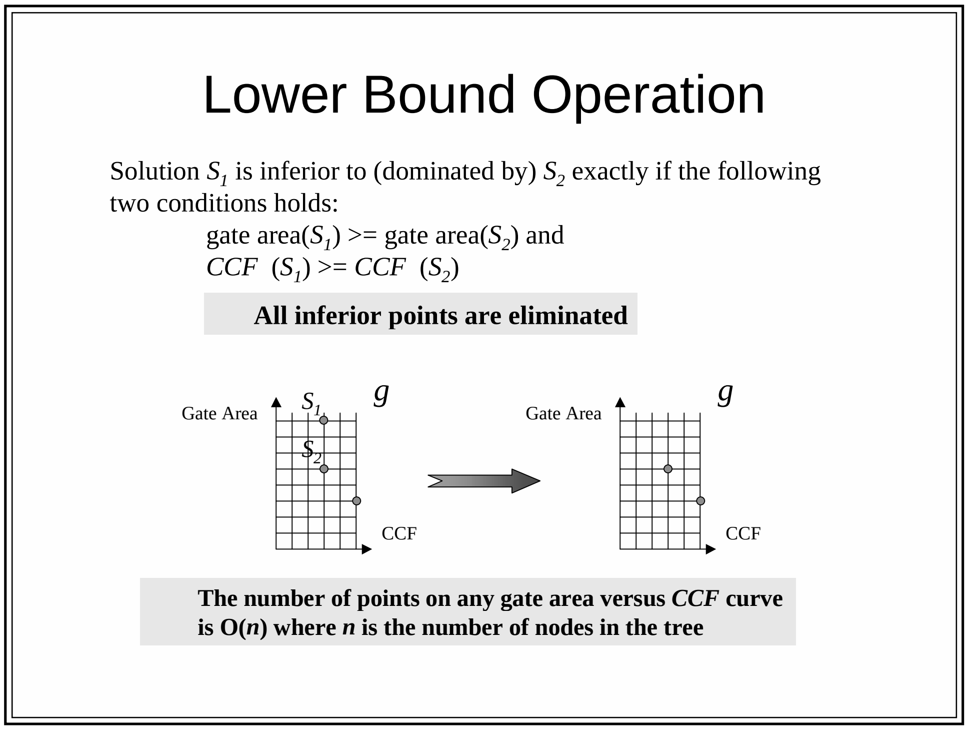
Solution: Use a gate area versus cut cost curve



Merge Operation



The runtime of the merge operation is $O\left(\sum_{i=0}^{k} N_i\right)$

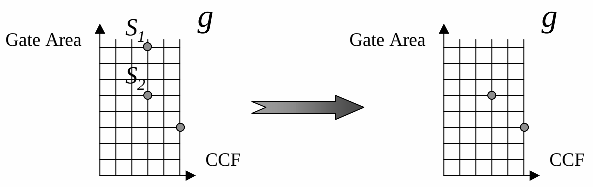


Lower Bound Operation

Solution S_1 is inferior to (dominated by) S_2 exactly if the following two conditions holds:

gate area(S_1) >= gate area(S_2) and
CCF (S_1) >= *CCF* (S_2)

All inferior points are eliminated



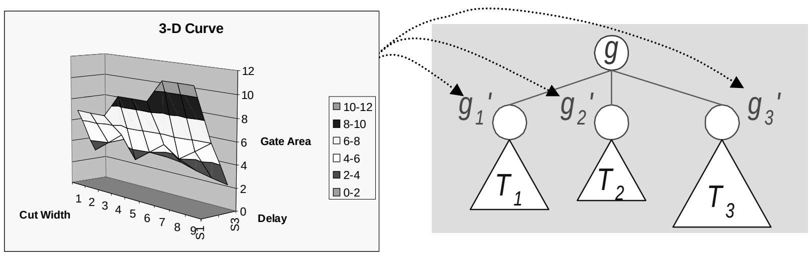
The number of points on any gate area versus *CCF* curve is O(*n*) where *n* is the number of nodes in the tree

SiMPA-E

```
SiMPA-E(N,L)
INPUT:    a tree network N, a library L
OUTPUT: a mapped and linearly placed network
1. Decompose N
2. Perform a reversed depth-first-search from PIs to the PO
3. For each node n in the reversed DFS order
4.    For every match m of node n
5.       For every point $p_0$ on the curve of $input_0$
6.          Find point $p_i$ on the curve of $input_i$ with MIN(gateArea) and
            $CCF(g_i)<=CCF(g_0)$
7.          $g = \sum_{\forall i} gateArea\,(p_i)$
8.          $c = ExactMincutPlace(m, p_0, p_1,\ldots)$
9.          Add <g, c> as a point in curve in n
10.    Prune inferior points on the cut width curve of n
11. Find the best solution according to the cost function from the (only)
    primary output, and recursively select solutions for all of its inputs
```

SiMPA-D

- Combines KA and LA
- Optimizes for total (gate plus wiring) delay
- Uses a gate area, *CCF*, and total delay 3-D curve

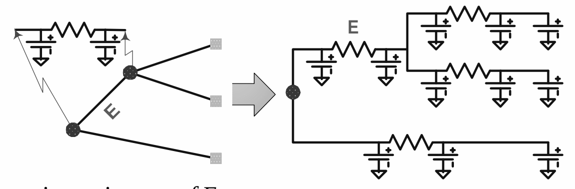


Delay Equations

Gate Delay Calculation:

$$\boxed{GateDelay = Slew \times (K_1 + K_2 \times load) + K_3 + K_4 \times load}$$

Wire Delay Calculation:



***r**$_E$: wire resistance of E*
***c**$_E$: wire capacitance of E*
***C**$_E$: total capacitance rooted at E*

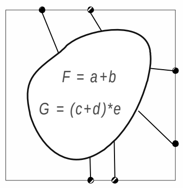
$$Delay\,(E) = r_E \times (\frac{c_E}{2} + C_E)$$

Experimental Results

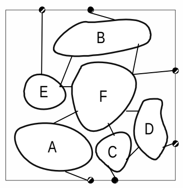
		Conventional					SiMPA-E			Area Ratio	Delay Ratio
	Gate Area	Cut	Delay	Total Area		Gate Area	Cut	Delay	Total Area		
tree6	7260	3	1.71	10428		7260	2	1.82	9372	89.87%	106.43%
tree8	10054	3	0.89	14441		10347	2	0.81	13357	92.49%	91.01%
tree16	17732	4	1.08	28049		18002	2	1.23	23239	82.85%	113.89%
tree20	30536	5	1.55	52744		31224	3	1.62	44849	85.03%	104.52%
tree32	38665	6	1.81	72409		39149	4	1.88	61927	85.52%	103.87%
tree48	66396	7	2.58	133999		68432	4	2.38	108247	80.78%	92.25%
										86.09%	101.99%

		Conventional					SiMPA-D			Area Ratio	Delay Ratio
	Gate Area	Cut	Delay	Total Area		Gate Area	Cut	Delay	Total Area		
tree6	9482	3	1.55	13620		10103	2	1.22	13042	95.76%	78.71%
tree8	13444	4	0.75	21266		15154	3	0.61	21767	102.35%	81.33%
tree16	22304	5	0.98	38525		19098	4	0.65	30210	78.42%	66.33%
tree20	51030	6	1.03	95565		54342	5	0.78	93863	98.22%	75.73%
tree32	48468	6	1.27	90767		50015	5	0.89	86390	95.18%	70.08%
tree48	90342	7	1.92	182327		85796	6	1.47	160673	88.12%	76.56%
										93.01%	74.79%

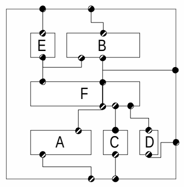
FPD-SiMPA



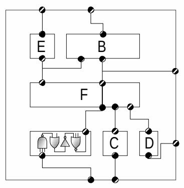
Input Circuit



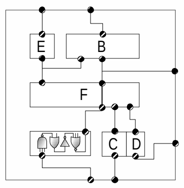
Tree Clustering



Floorplanning, Global Routing and Budgeting

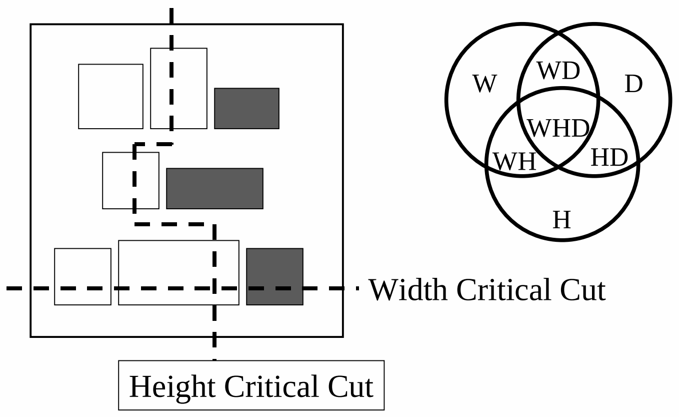


Technology Mapping and Placement

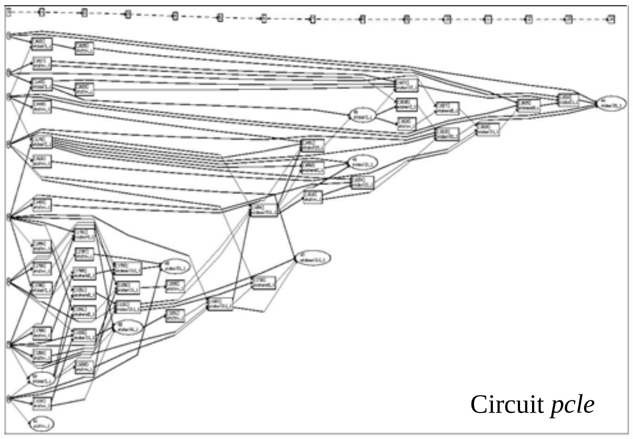


Final Layout Generation

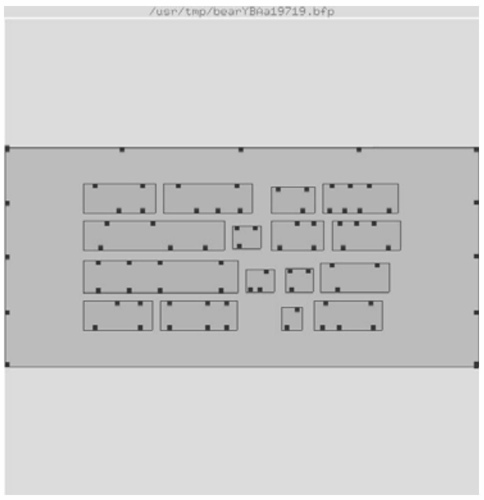
Critical Cluster Selection



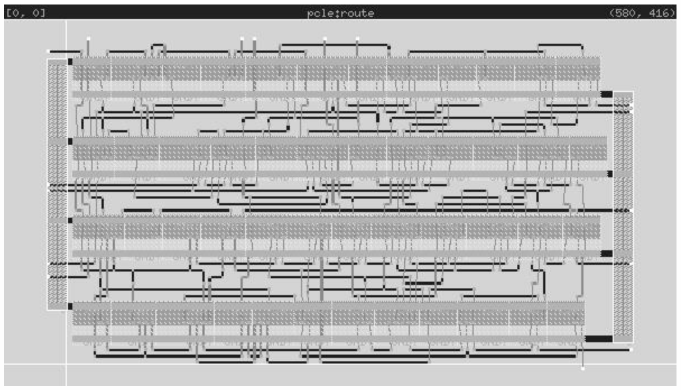
An Example


Circuit *pcle*

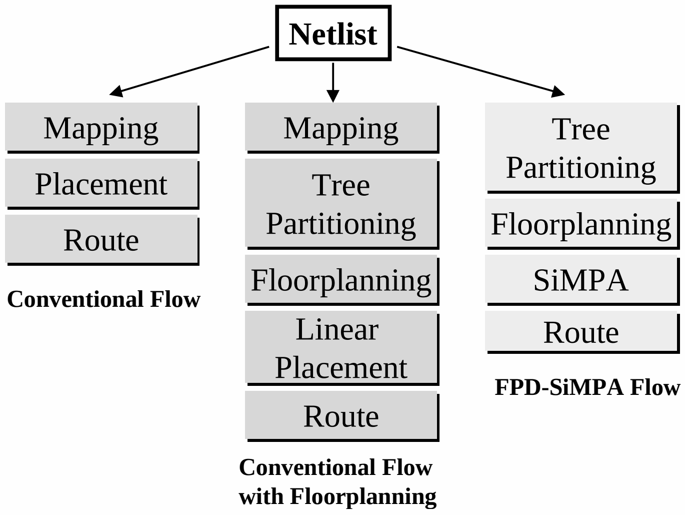
BEAR-FP View

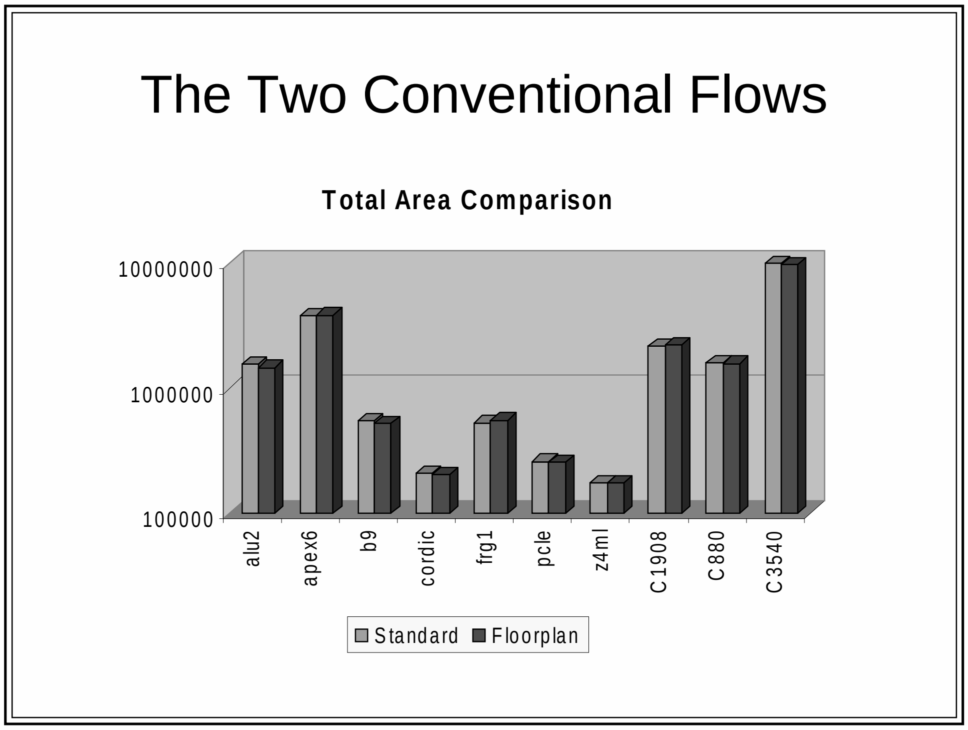


Post-layout View

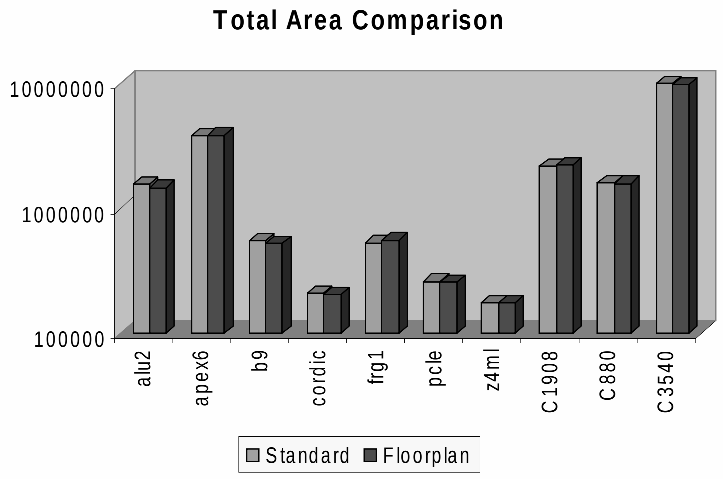


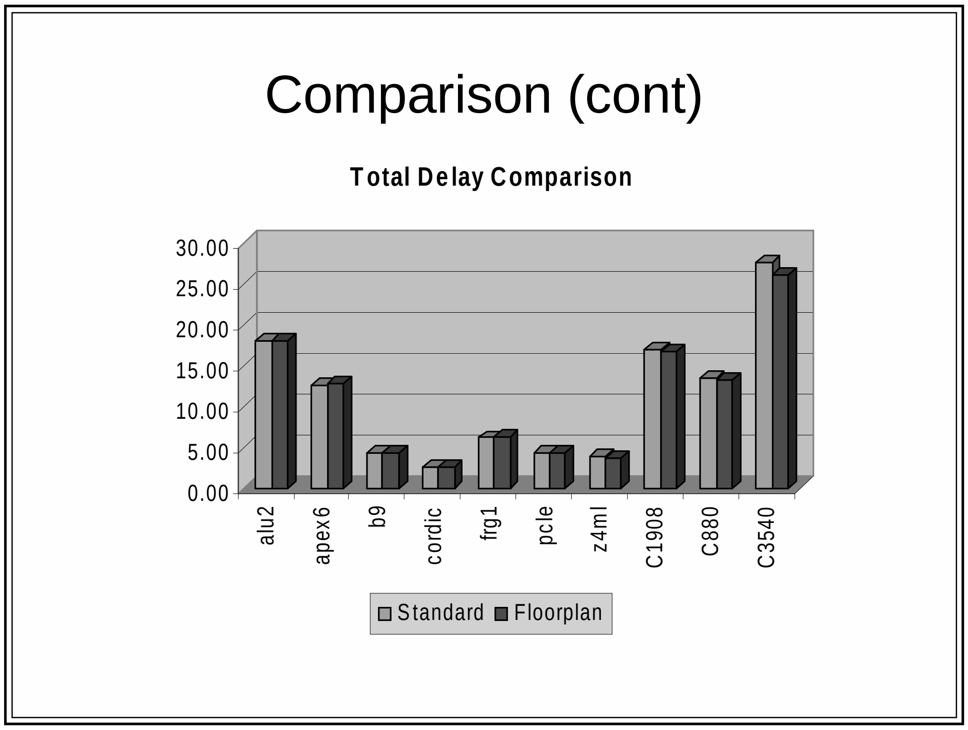
Experimental Setup



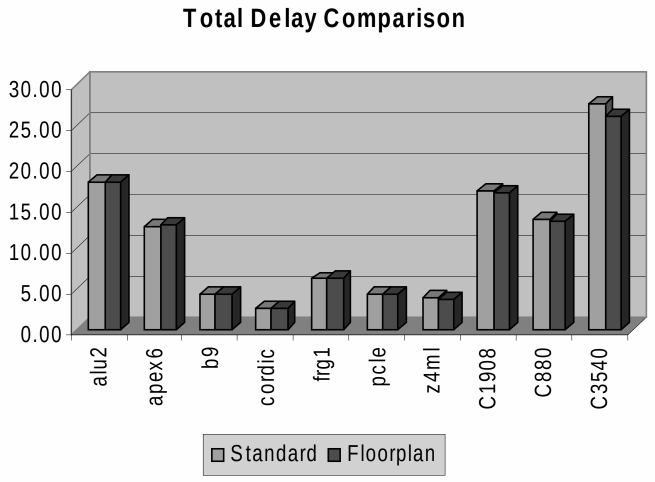


The Two Conventional Flows

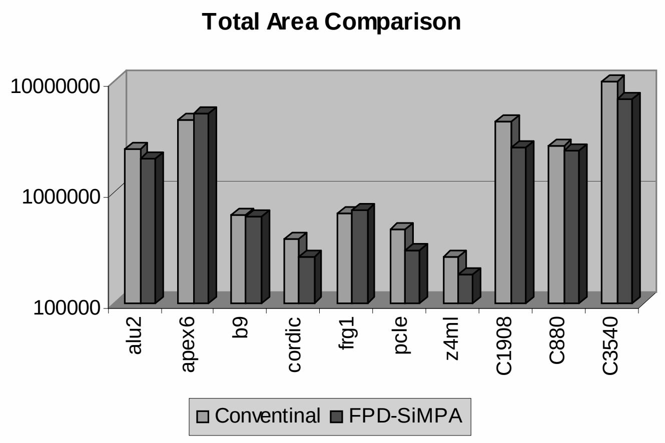




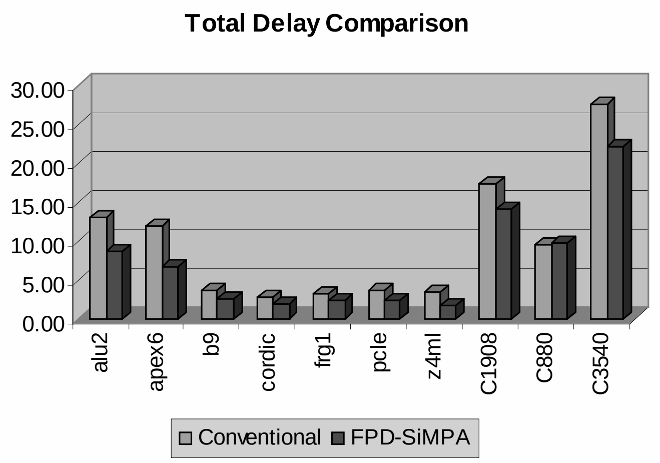
Comparison (cont)



Conventional vs. FPD-SiMPA Flow

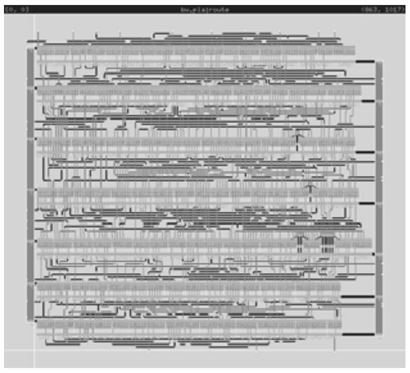


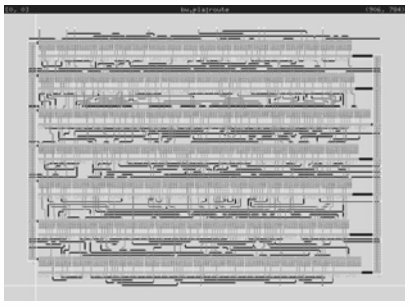
Comparison (cont)



Post-layout Views

Circuit *bw*



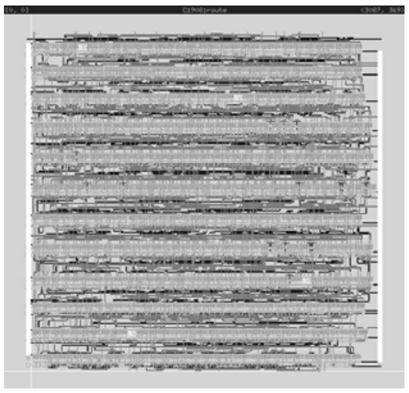


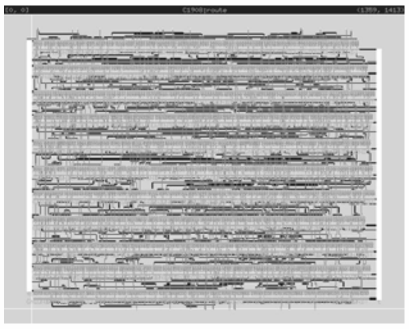
FPD-SiMPA Flow

Conventional Flow

Views (cont)

Circuit *C1908*





FPD-SiMPA Flow

Conventional Flow

Conclusion

- SiMPA optimally solves the simultaneous technology mapping and linear placement problem for tree-structured circuits
- FPD-SiMPA combines floorplan-driven flow and SiMPA to produce high quality solutions for general circuits
- Future work will focus on developing non-tree circuit partitioning, direct two-D placement, and global wire planning.