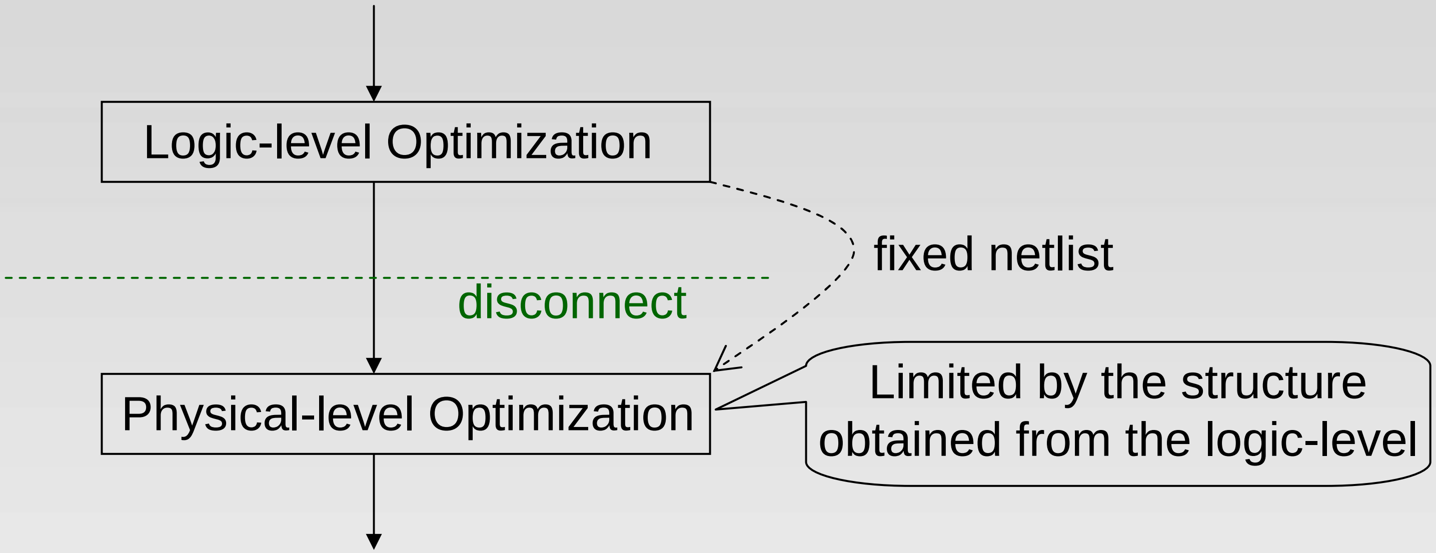
University of Illinois at Chicago
Department of Computer Science

A Framework for Layout-Level Logic Restructuring

Hosung Leo Kim
John Lillis

Motivation: Logical-to-Physical Disconnect



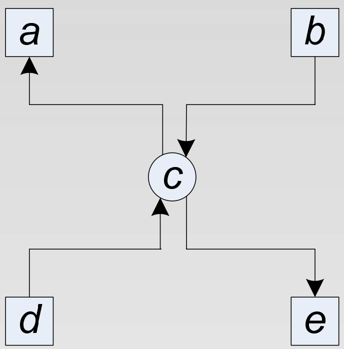
- Performance is determined largely by physical-level *Interconnect delay.*
- Problem: timing optimization at logic-level ≠ actual performance.

Past Layout-Driven Restructuring Work: Replication Based

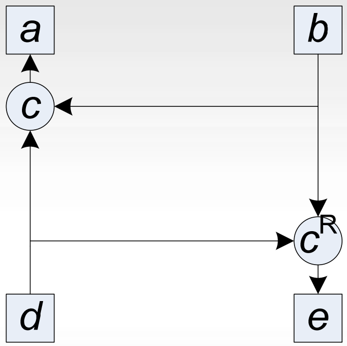
- Basic Operations:
  - Gate Splitting
  - Fanout Partitioning;

⇨ Enables "Path Straightening"

  - [Schabas, Brown. ISFPGA03]
  - [Beraudo, Lillis. DAC03]
  - [Hrkic, Lillis, Beraudo. TCAD06]
  - [Chen, Cong. ISFPGA05]



(a) Inheritantly non-monotone paths.



(b) Timing-improved with replication and relocation of cell *c*.

Limitation of Logic Replication

- While interconnect delay can be significantly reduced, the LUT-depth of a path remains unchanged.
- The LUT-depth is typically determined by a technology mapper which does not have an accurate view of critical paths.

⇨ Candidate: Remapping

Other Work

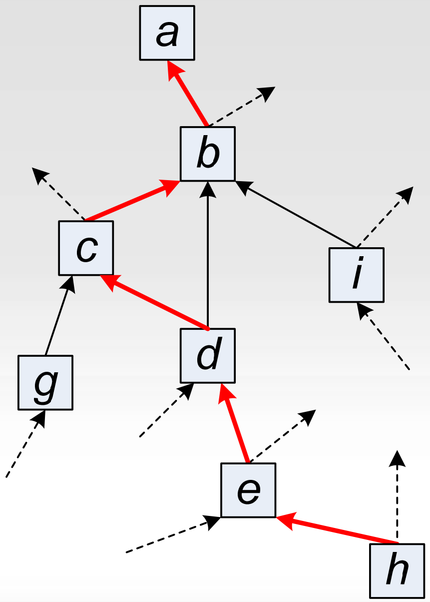
- Redundant Wires (e.g., [Chang, Cheng, Suaris, Marek-Sadowska. DAC00])
  - *rewire connections while keeping logical equivalence.*
  - *Predictable, but optimization scope limited*
- [Lin, Jagannathan, and Cong. ISFPGA03]
  - *Remap based on placement-level timing analysis*
  - *Significant restructuring, but placement of remapped cells determined by initial placement (not simultaneous).*
- [Singh and Brown. Integration07]
  - *Shannon's expansion / precomputation*
  - *Allows late signals to skip logic levels, but relatively local in nature*

Objectives

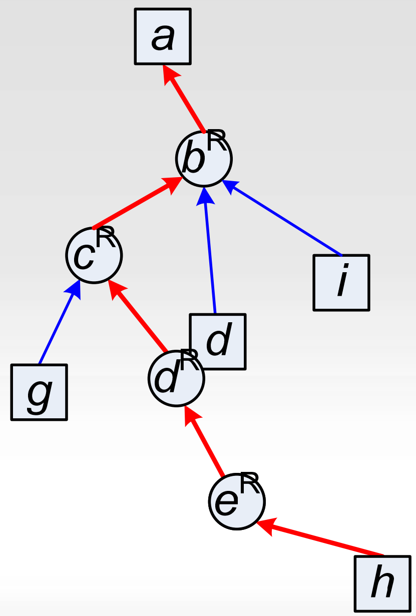
- Overcome limitations of basic replication (e.g., fixed LUT-depth)
- Large and flexible remapping space
- Explicitly account for placement freedom of remapped LUTs
- Tight coupling with placement

Components of Approach (FPGA Domain)

Placement-Level Static Timing Analysis → **Timing-Critical Fan-in Cone Extraction** → **Induce Replication Tree [Hrkic,TCAD06]** →



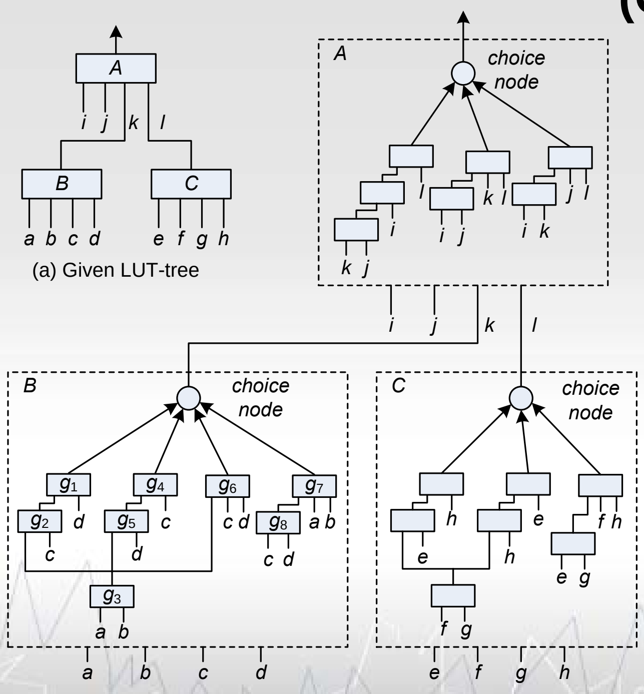
(a) Sub-circuit containing the slowest path



(b) Induced replication tree

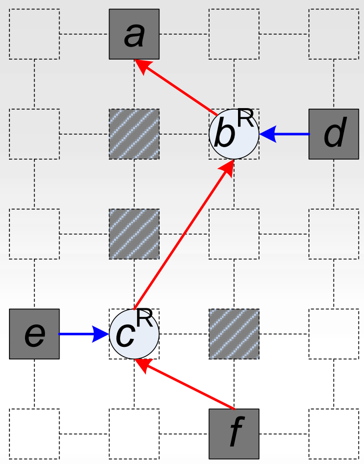
Components of Approach (cont'd)

Replication tree → **Recursive, Exhaustive Ashenhurst LUT Decomposition** → Subject Graph (Choice Tree) → **Mapper and Embedder (Dynamic Programming)** → **Legalizer**

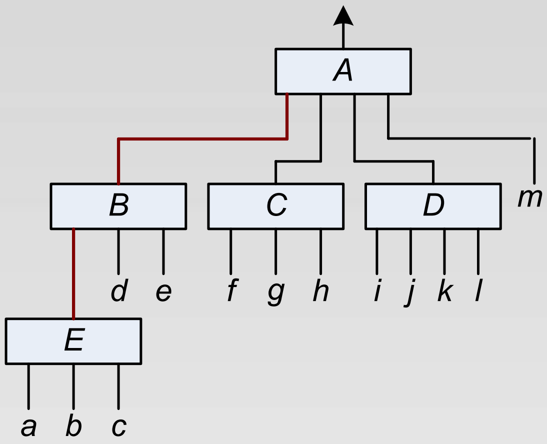


(a) Given LUT-tree

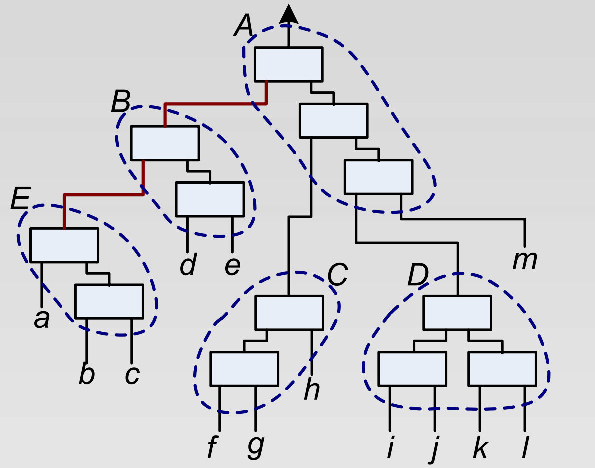
(b) Choice tree



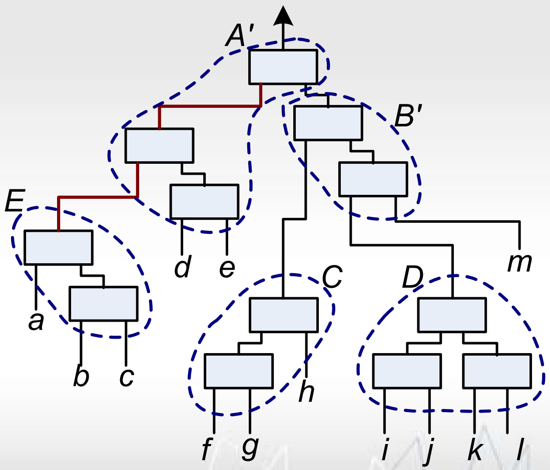
Remapping Example



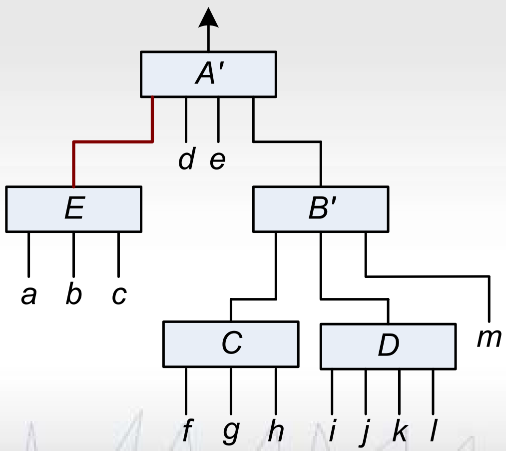
(a) Given LUT-tree



(b) “Mini-LUT” tree after LUT-decompositions



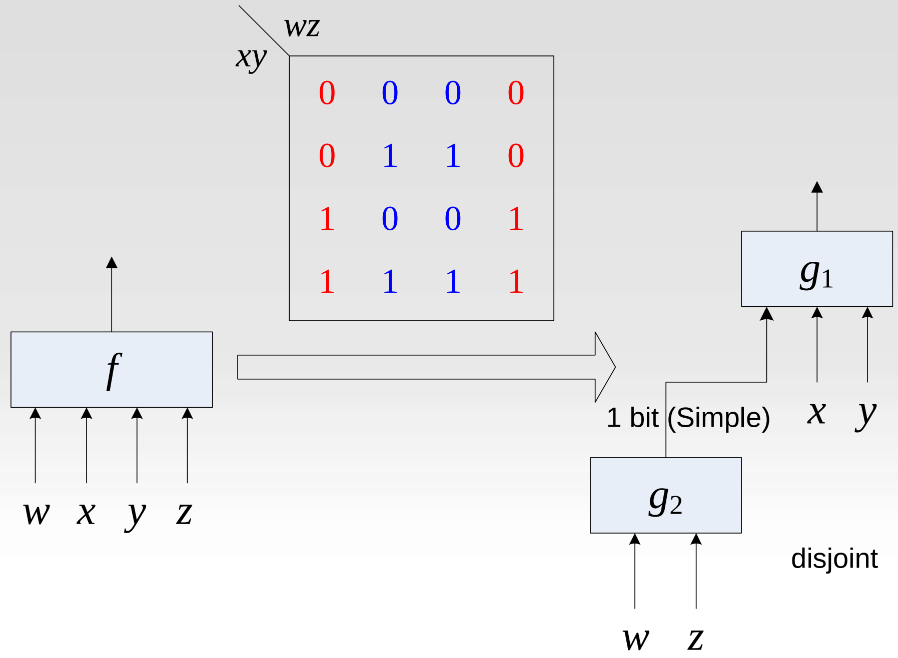
(c) Alternative mapping

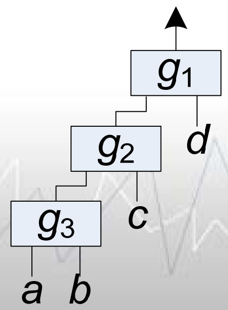


(d) Corresponding LUT-tree

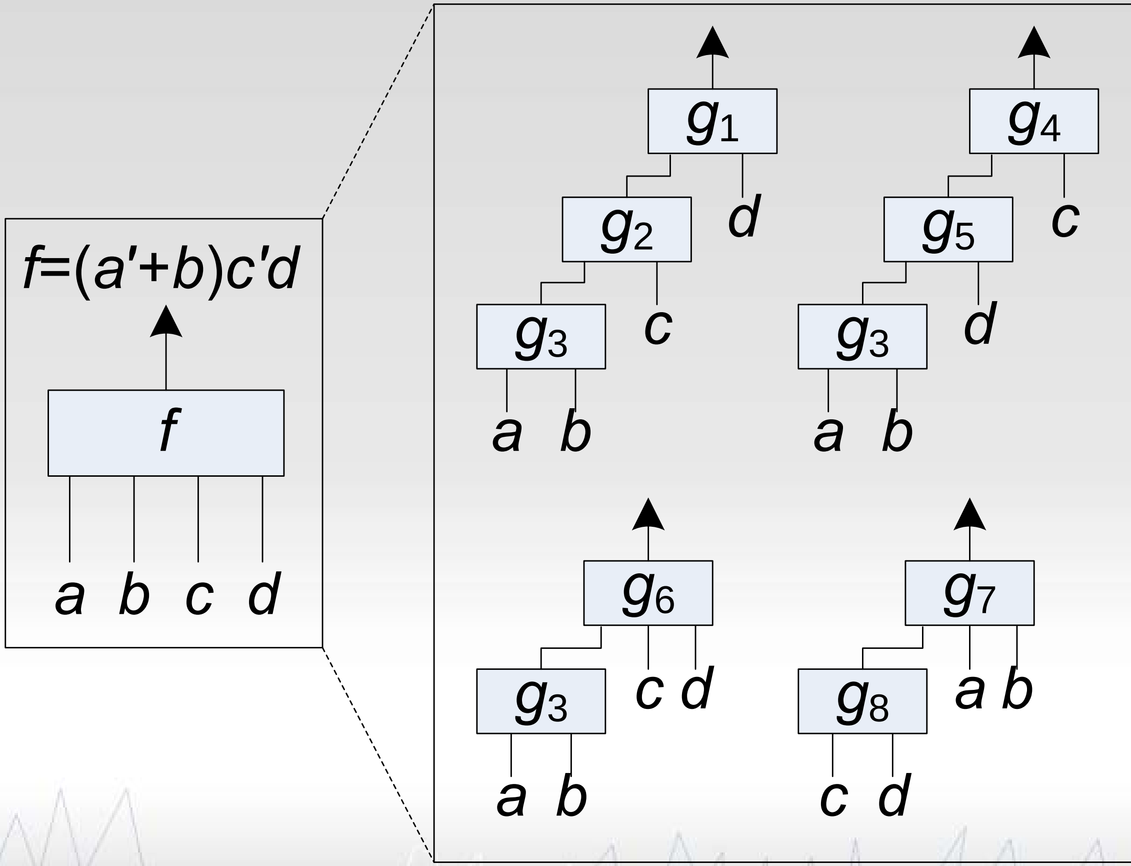
Functional Decomposition

- Simple Disjoint Functional Decomposition
- Test for decomposability
  - Ashenhurst's theorem
- Recursively decompose



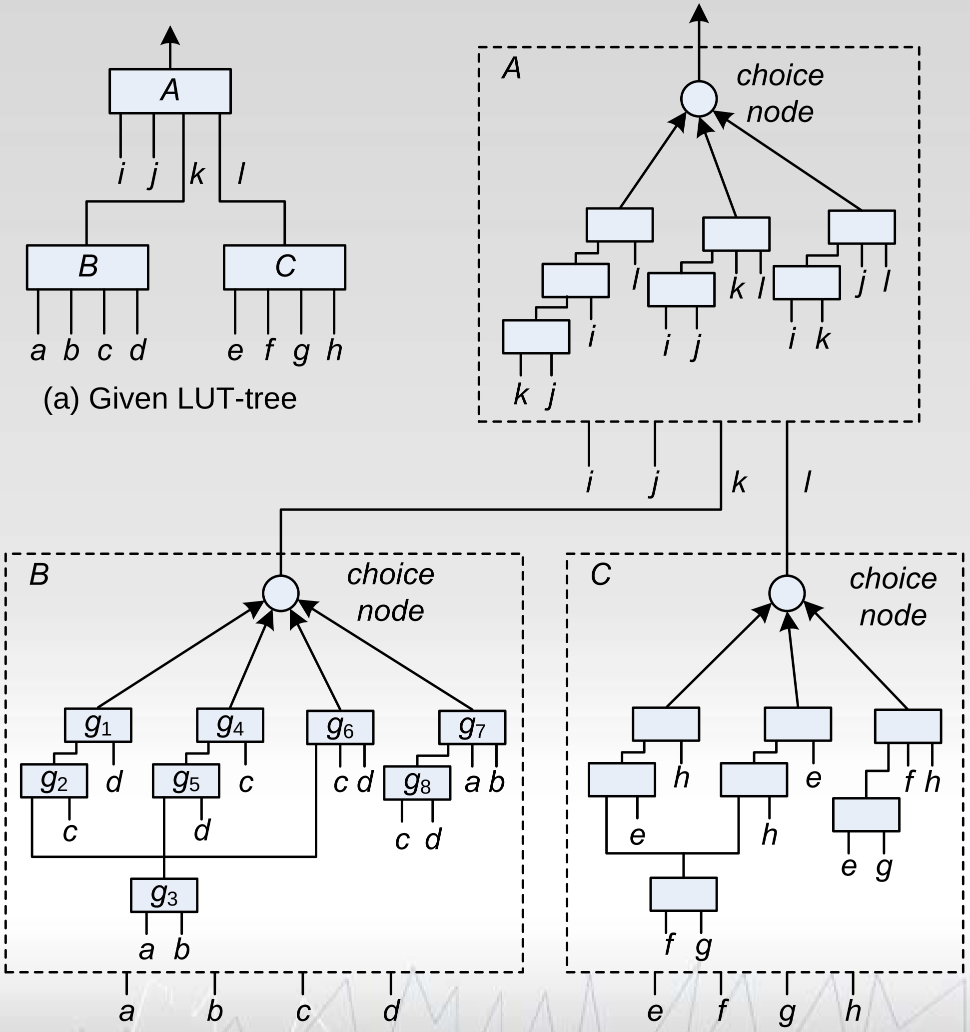


All Recursive Decompositions



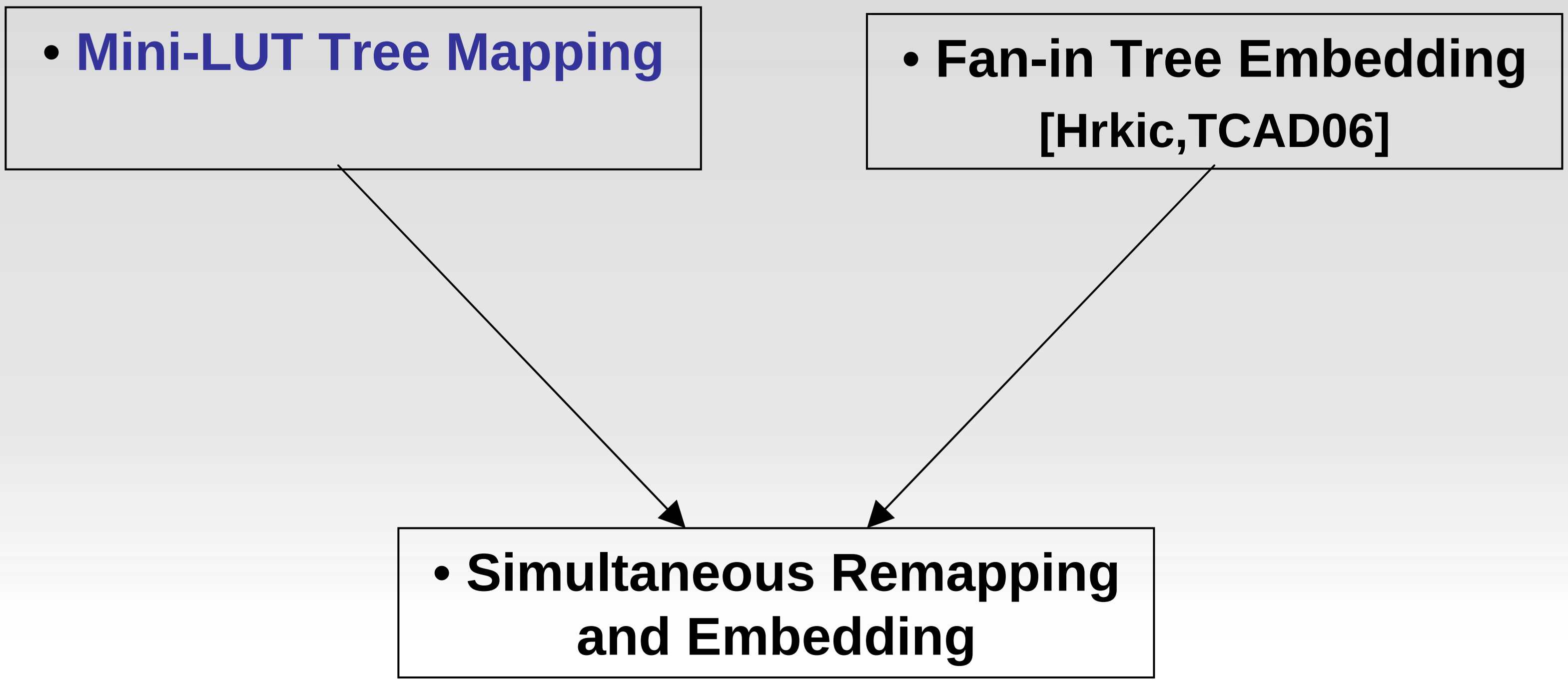
$g_1=(g_2)d$
$g_2=(g_3)c'$
$g_3=(a'+b)$
$g_4=(g_5)c'$
$g_5=(g_3)d$
$g_6=(g_3)c'd$
$g_7=(a'+b)(g_8)$
$g_8=c'd$

Choice Tree [Lehman,TCAD97]



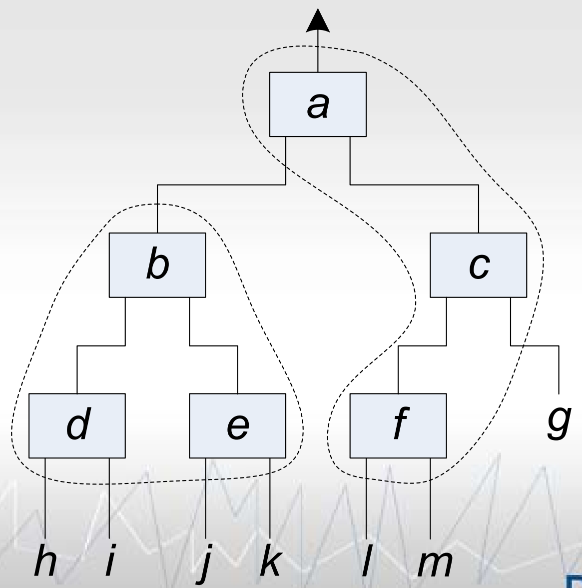
(b) Choice tree

Algorithm


• Mini-LUT Tree Mapping
• Fan-in Tree Embedding
[Hrkic,TCAD06]
• Simultaneous Remapping
and Embedding

Logic Remapping Formulation

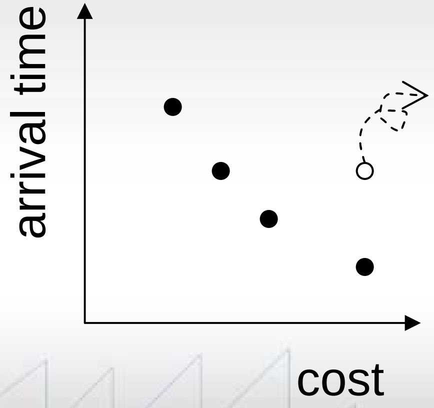
- Formulation
  - Given a "mini-LUT" tree and arrival time at the leaves,
  - map the tree to *K*-input LUTs minimizing cost subject to an arrival time constraint at the root.



Solution Signature

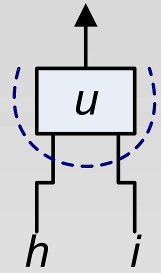
- (c,a)
  - for a sub-tree rooted *u*, a solution is characterized by two parameters:
    - *cost* of the embedding (and remapping) of a sub-tree.
    - *arrival time* at *u*.

- Dominance Relation
  - (*c*,*a*) is not dominated by (*c*',*a*') when *c* is better than *c*' or *a* is better than *a*'.

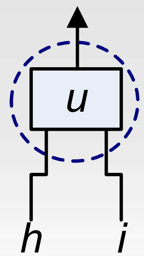


Solution Sets

- $S_i^J[u] = \{(c,a)\}$
  - u: signal produced by root LUT
  - i: # inputs of root LUT
  - c: # LUTs in subtrees
  - a: the latest among the fan-ins.

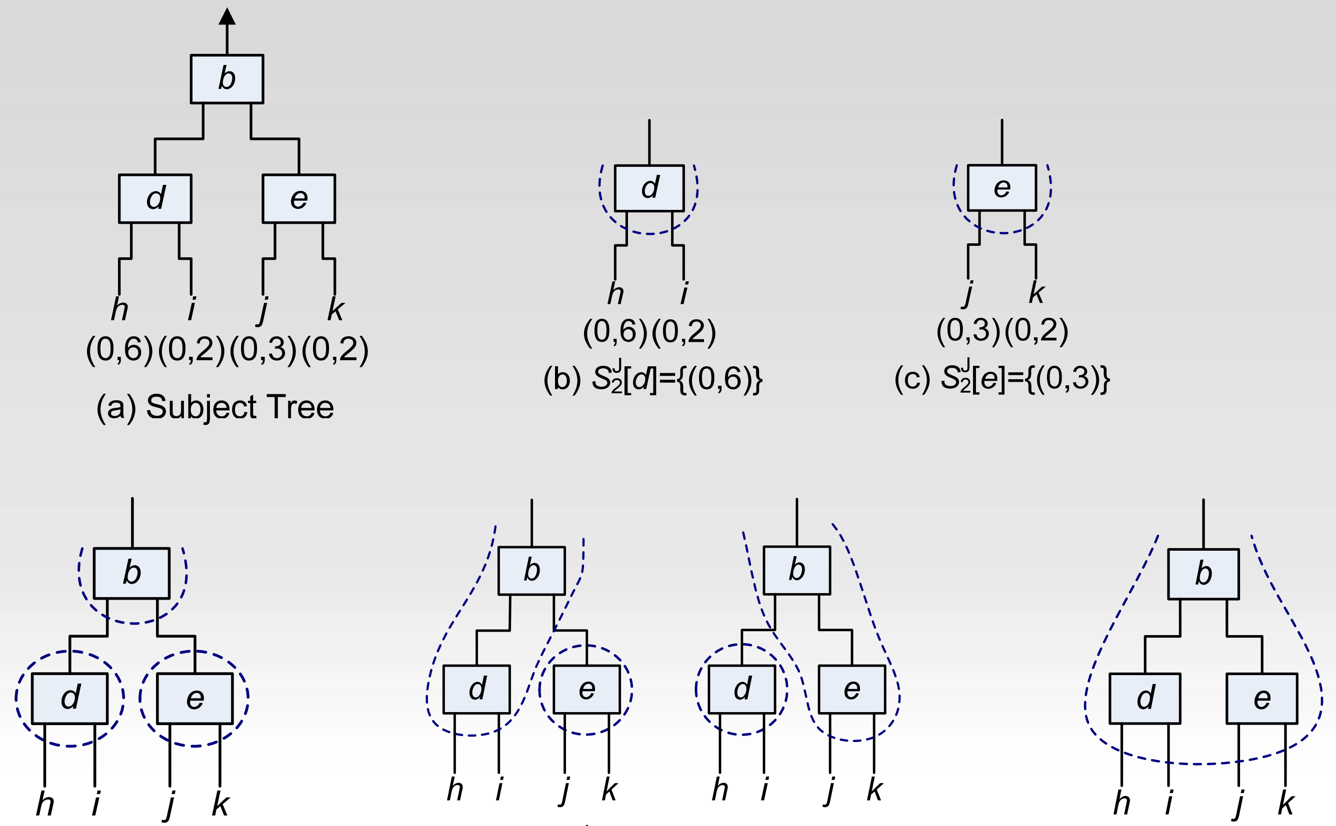


- $S_i[u]$
  - "finalized" solution from $S_i^J[u]$.
  - c: # LUTs in subtrees + 1
  - a: the root LUT included.



- $S[u]$
  - non-dominated_sol($S_2[b]$, … , $S_K[b]$)

$S_i^J[u]$ Example



(a) Subject Tree

(b) $S_2^J[d]$={(0,6)}

(c) $S_2^J[e]$={(0,3)}

(d) $S_2^J[b]$={(2,7)} =join($S[d]$,$S[e]$)

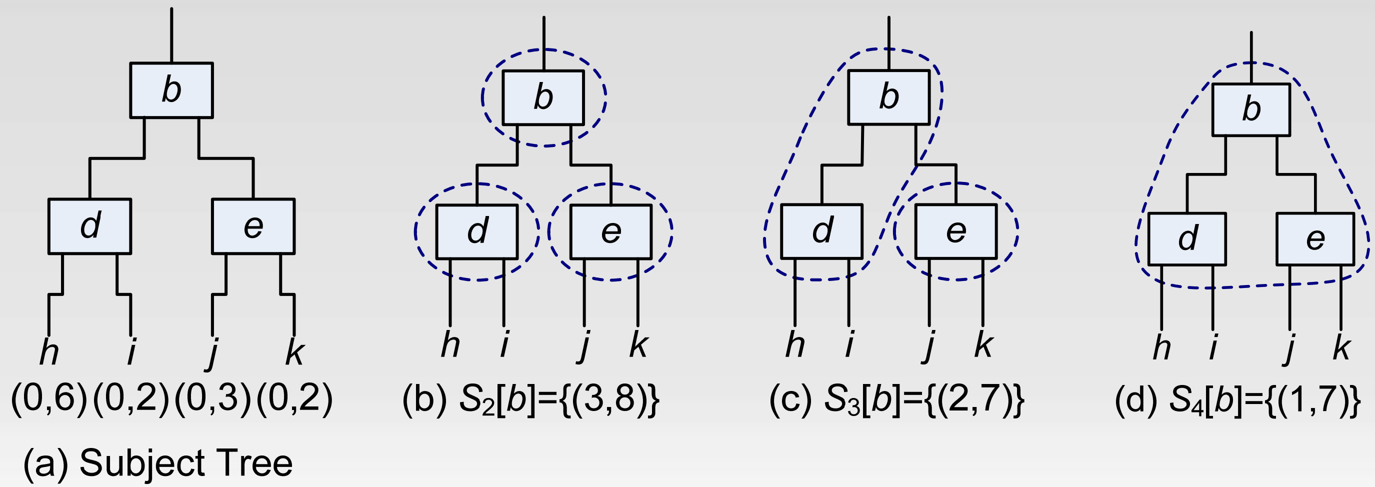
(e) $S_3^J[b]$={(1,6),~~(1,7)~~} =join($S_2^J[d]$,$S[e]$)$\cup$join($S[d]$,$S_2^J[e]$)

(f) $S_4^J[b]$={(0,6)} =join($S_2^J[d]$,$S_2^J[e]$)

For simplicity:
one LUT = one unit cost
one LUT = one unit delay

$S_i[u]$ and $S[u]$ Example

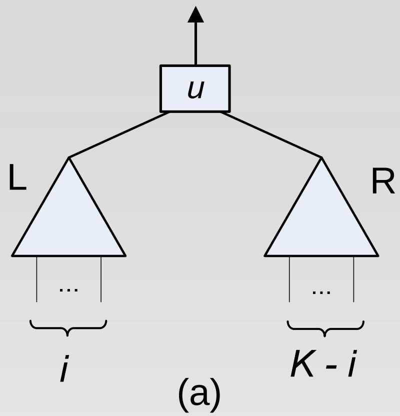
- $S_i[b]$



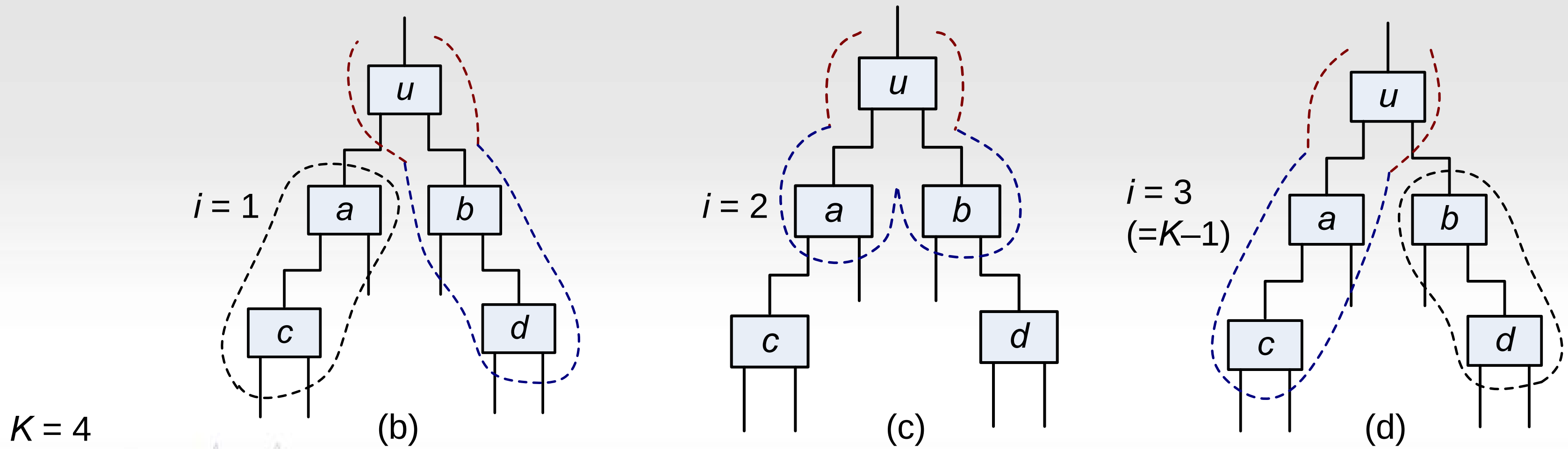
- $S[b]$ = non-dominated_sol($S_2[b]$, … , $S_K[b]$)

  = {(1,7)}

Computation of $S_i^J[u]$

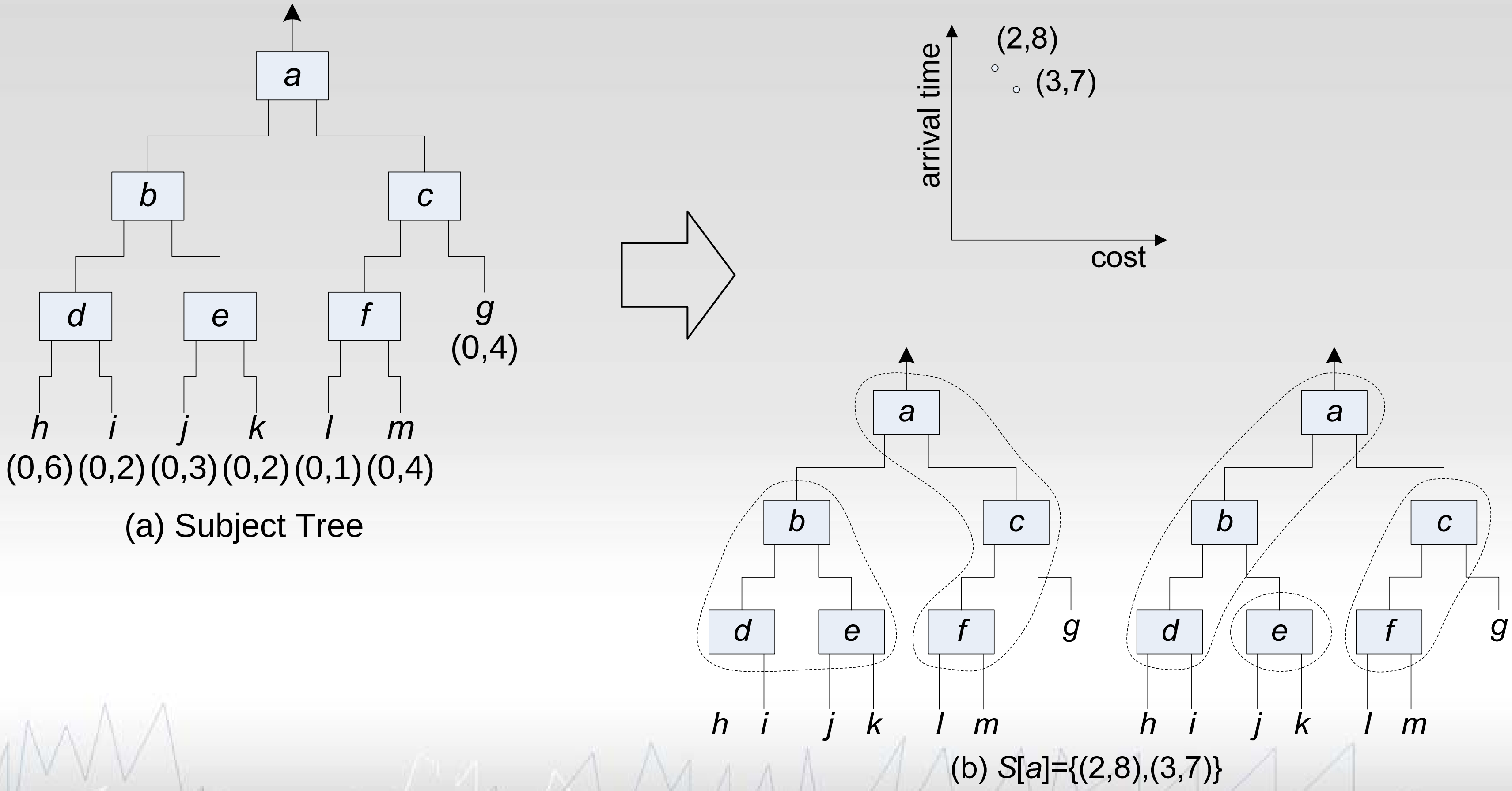


i = 1, no collapsing of *u* and L
i = K-1, no collapsing of *u* and R
Otherwise, collapsing of *u*, L, and R.

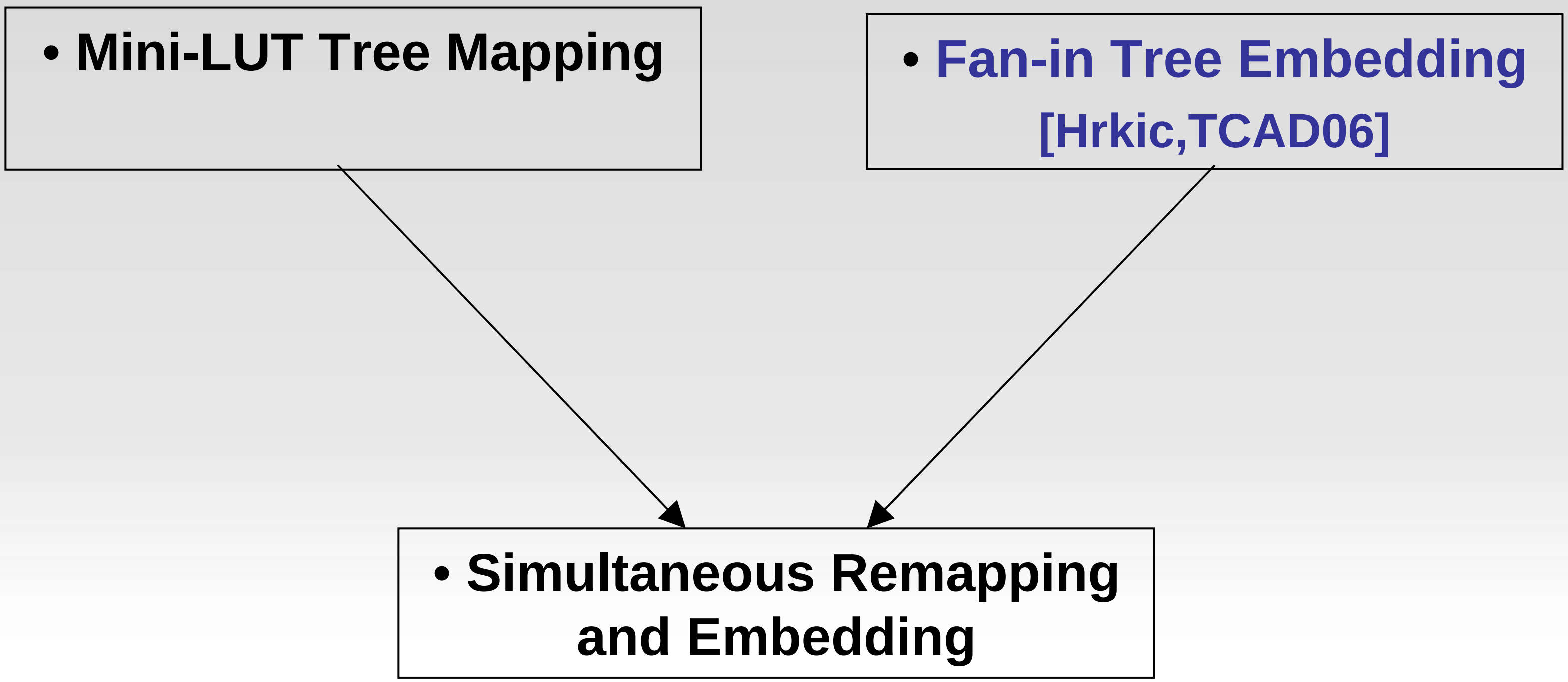


$$S_4^J[u] = \text{join}(S[a], S_3^J[b]) \cup \text{join}(S_2^J[a], S_2^J[b]) \cup \text{join}(S_3^J[a], S[b])$$

Remapping Algorithm Example

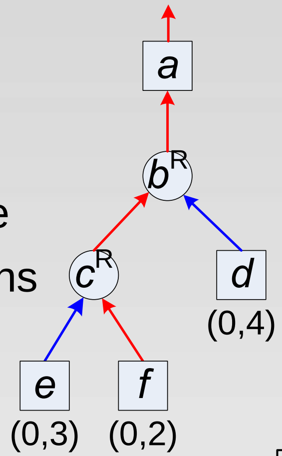


Algorithms

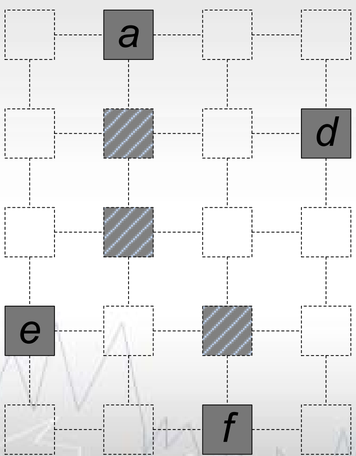

• Mini-LUT Tree Mapping
• Fan-in Tree Embedding [Hrkic,TCAD06]
• Simultaneous Remapping and Embedding

Tree Embedding [Hrkic,TCAD06]

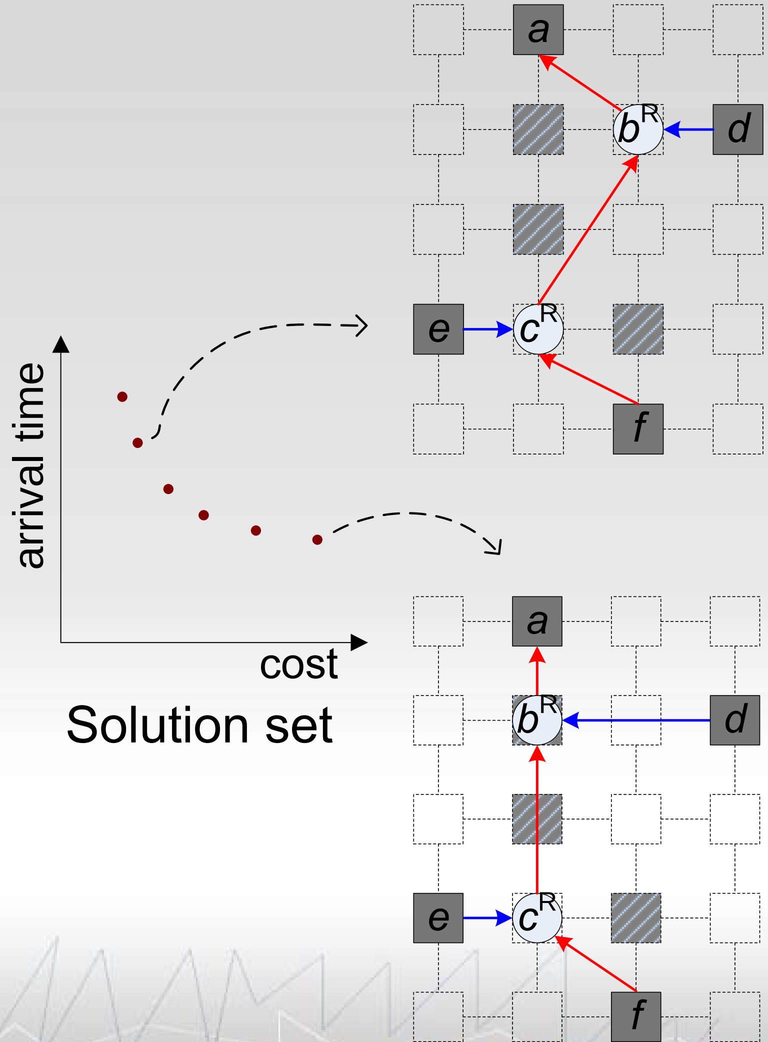
topology
arrival time
pin locations



target layout graph
cost metrics



Embedding Algorithm



UIC University of Illinois at Chicago
Department of Computer Science

Algorithms

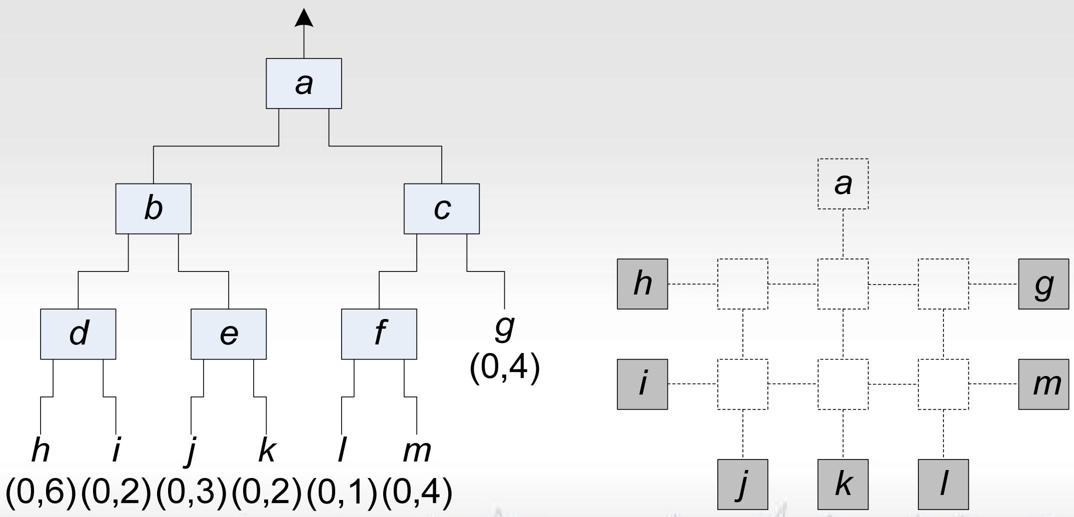
- **Mini-LUT Tree Mapping**

- **Fan-in Tree Embedding**
  [Hrkic,TCAD06]

- **Simultaneous Remapping and Embedding**

Simultaneous Remapping and Embedding

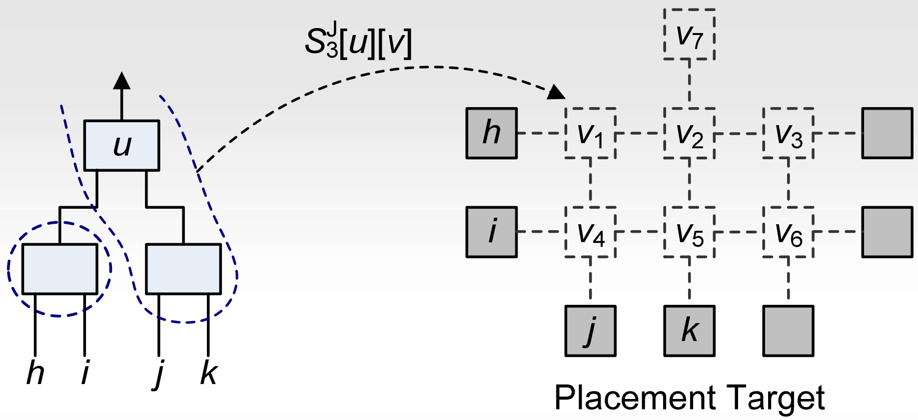
- Formulation
  - Given a "mini-LUT" tree with fixed leaves and root, and arrival time at the leaves, a target layout graph
  - Simultaneously map the tree to *K*-input LUTs and embed.



(a) Subject Tree (b) Placement Target

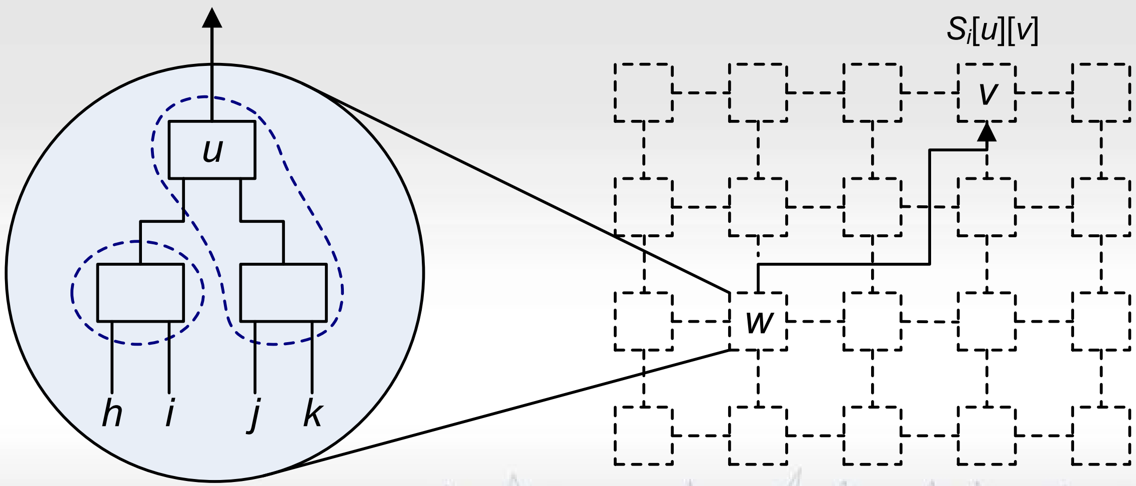
Solution Set $S_i^J[u][v]$

- The remapped root produces signal *u* and <u>is placed at *v* in the target layout graph.</u>



Solution Set $S_i[u][v]$

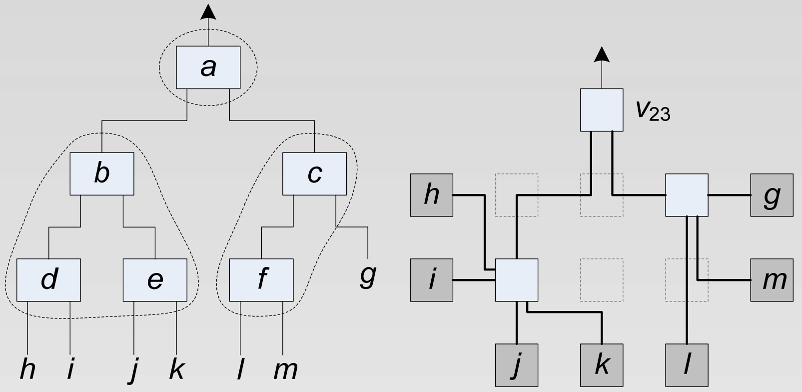
- Solutions $S_i^J[u][w]$ are finalized and drives vertex *v* in the target layout graph.
- Computed by shortest weight-constrained path algorithm.



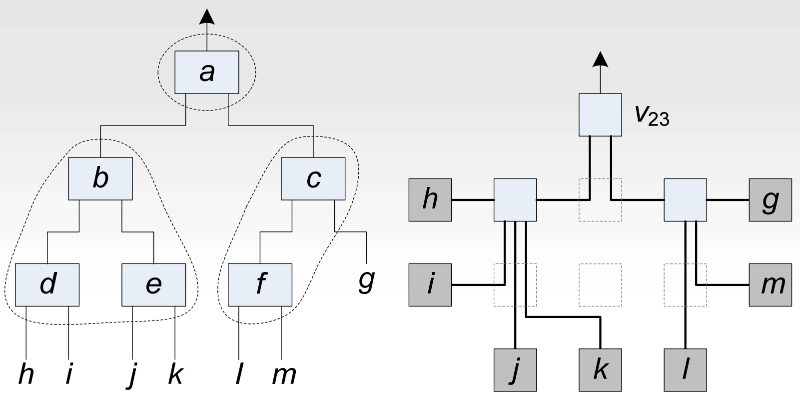
Solution Set $S[u][v]$

- $S[u][v] \leftarrow$ non-dominated-sol($S_2[u][v],\ldots,S_K[u][v]$)
- The best remapping regardless of the number of inputs at *v* in the target layout graph.

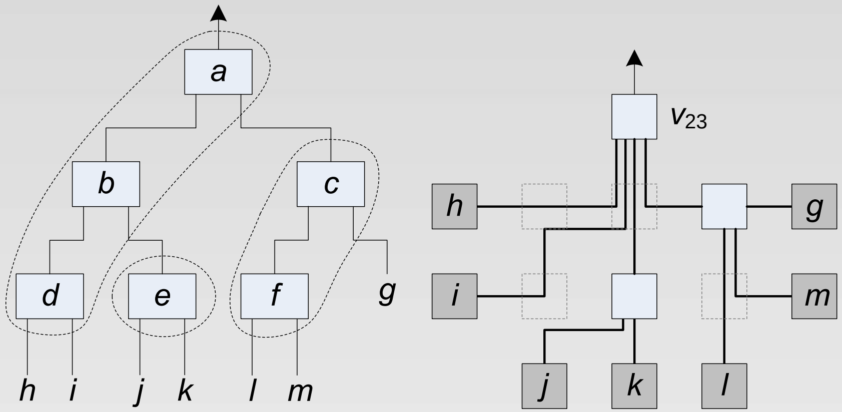
Simultaneous Remapping and Embedding Example



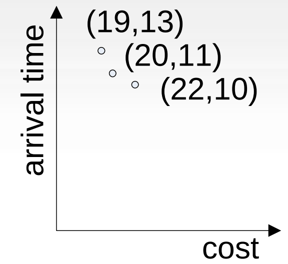
(a) $S_2[a][v_{23}]$={(19,13),(20,11)}



(b) $S_2[a][v_{23}]$={(19,13),(20,11)}



(c) $S_4[a][v_{23}]$={(22,10)}

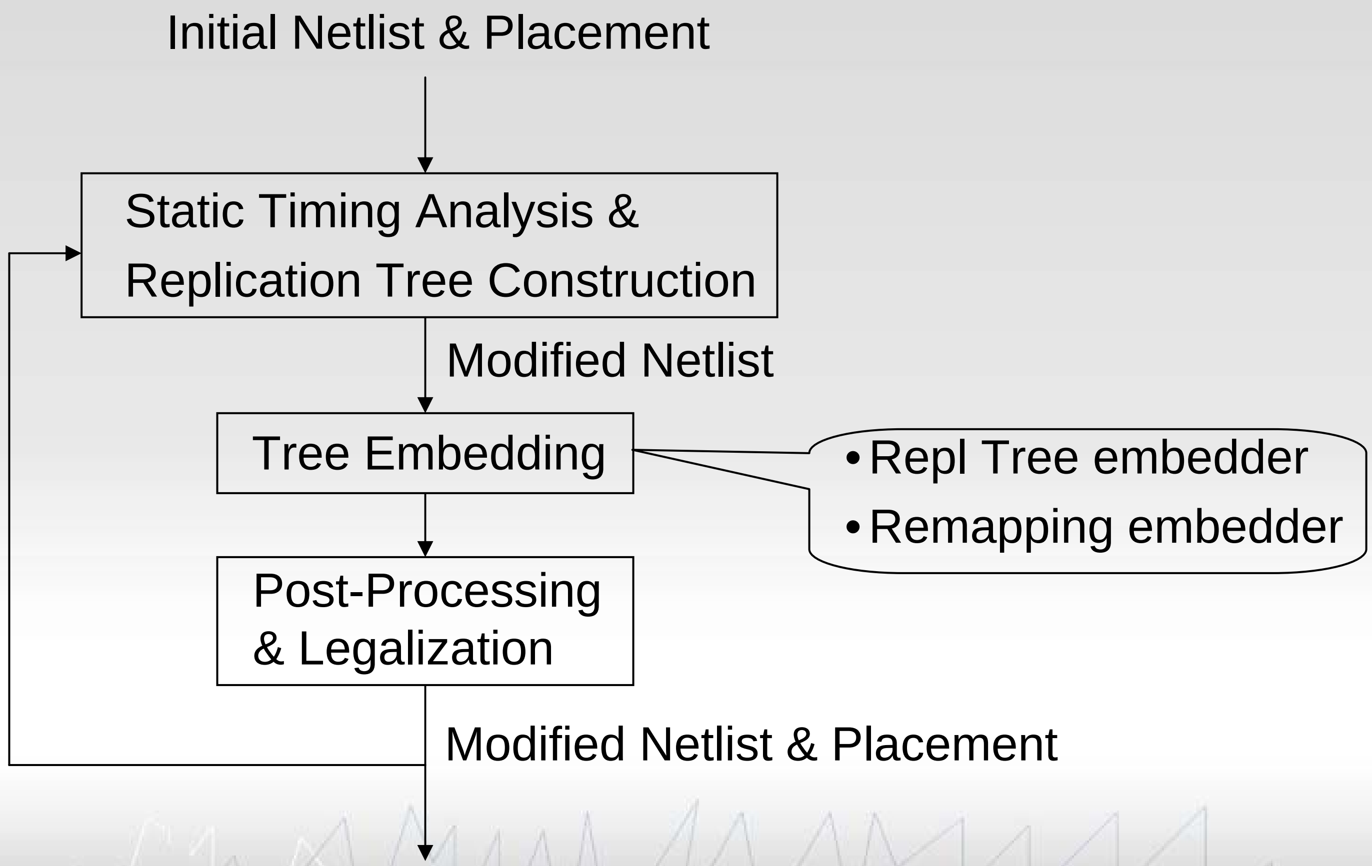


(c) $S[a][v_{23}]$={(19,13),(20,11),(22,10)}

Experiment

- Benchmarks
  - 20 MCNC benchmark circuits
  - At least 20% white space
- Comparisions
  - Timing-driven VPR placer
  - Replication Tree embedder
  - Arbor embedder [Kim,GLSVLSI06]
  - Remapping embedder
- Criteria of Interest
  - LUT depth
  - Clock period of circuits
- Different logic-level mappers and Stability effect of new algorithm

Optimization Flow

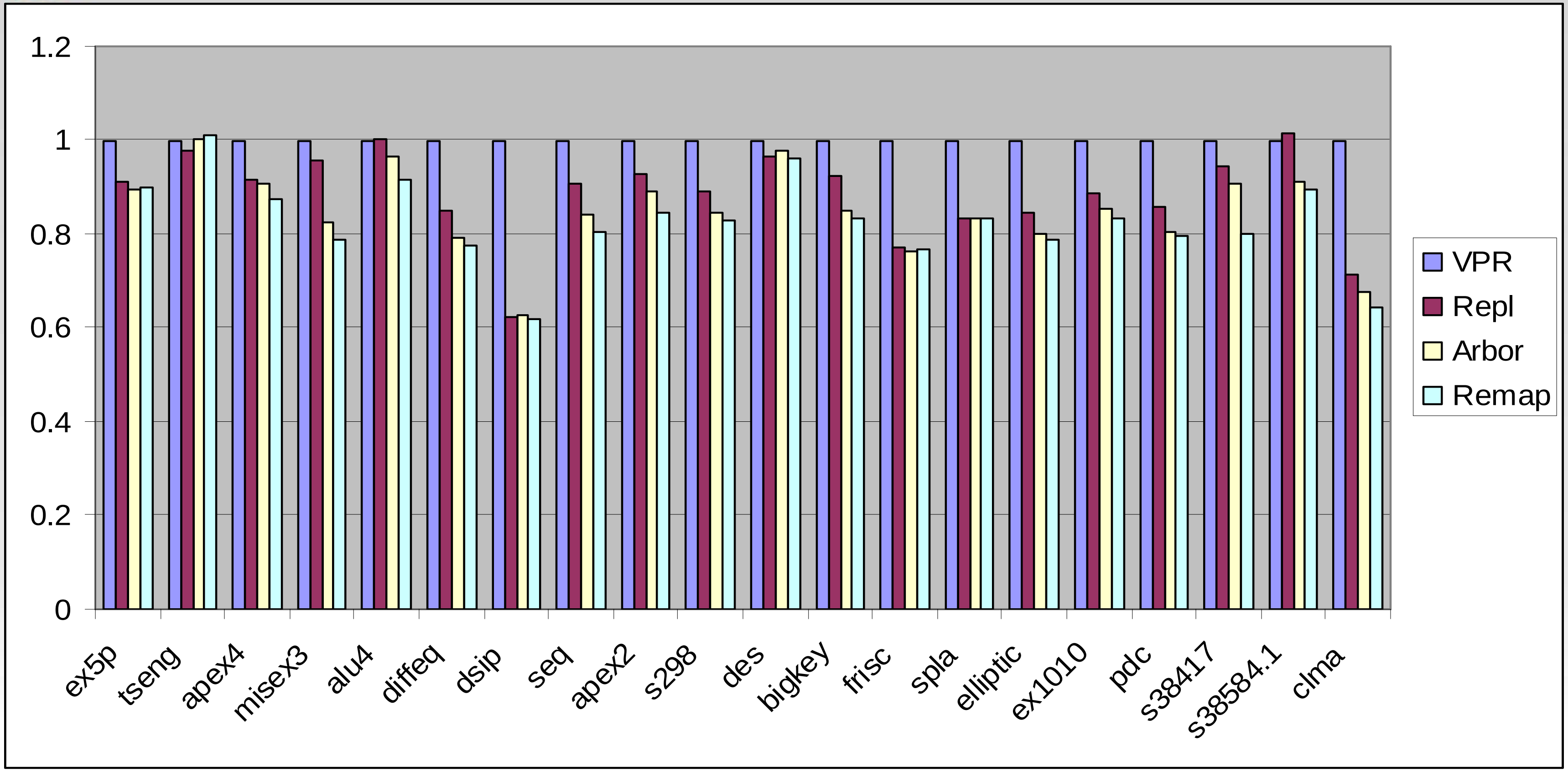


LUT Depth Changes

	Init. ckt		New ckt	
	ckt	Crit. Path	ckt	Crit. Path
ex5p	9	8	9	7
tseng	12	12	12	11
apex4	8	8	9	8
misex3	9	8	9	8
alu4	9	9	9	8
diffeq	**14**	**14**	**13**	**10**
dsip	8	5	8	4
seq	8	8	9	7
apex2	10	10	10	9
s298	**16**	**16**	**14**	**13**

	Init. ckt		New ckt	
	ckt	Crit. Path	ckt	Crit. Path
des	8	5	8	4
bigkey	5	4	5	4
frisc	23	20	22	22
spla	10	10	10	9
elliptic	17	15	18	15
ex1010	10	10	10	9
pdc	11	10	11	8
s38417	10	10	10	8
s38584.1	10	5	10	5
clma	**15**	**14**	**14**	**13**

Routed Clock Period



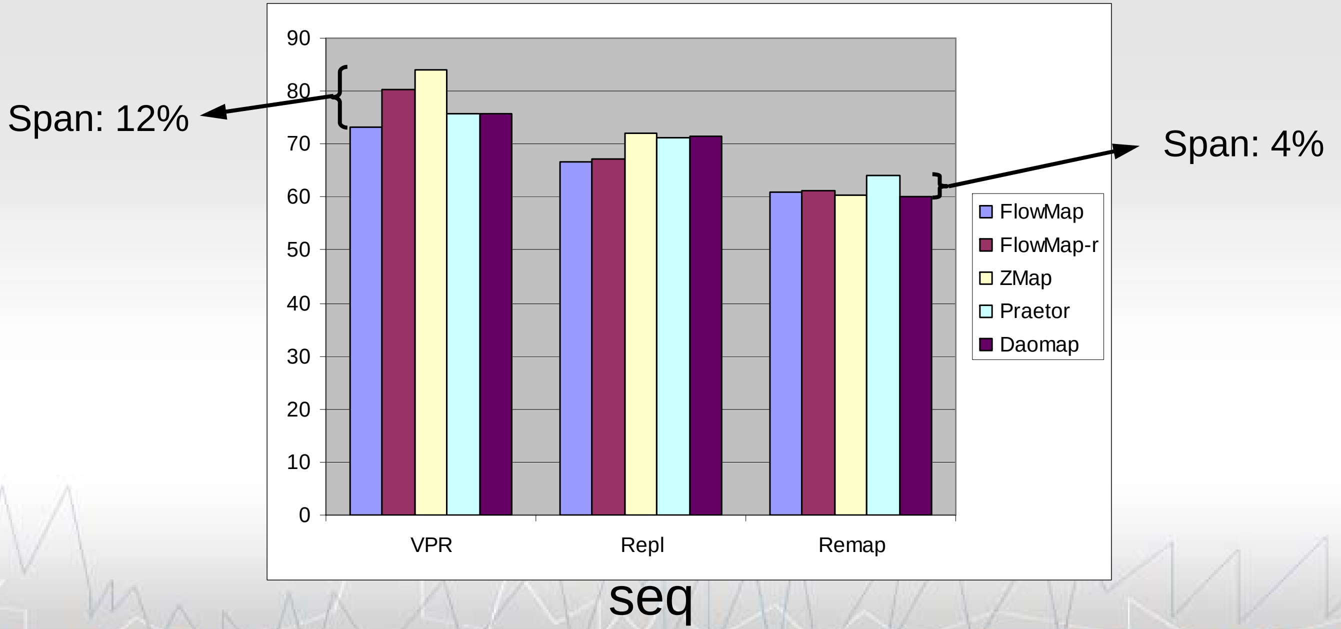
- Average Normalized Clock Period

	T-VPR	Repl	Arbor	Remap
Avg Delay	1	0.886	0.848	0.826

- Max reduction of REMAP vs Arbor 11.7%

Different Logic-level Mappers and Stability Effect of Remap

- FlowMap: optimal depth.
- FlowMap-r: relaxed depth.
- ZMap: optimal depth with simultaneous area minimization.
- Praetor: minimized area.
- Daomap



Summary

- Study of layout-level restructuring for interconnect optimization.
  - Functional Decomposition
  - Choice Tree
  - Remapping Algorithm
  - Simultaneous remapping and embedding
- Experimental Result
  - Average 17% reduction on clock period compared with T-VPR.

Thank You!