Logic Decomposition and Technology Mapping - Lehman-Watanabe Method

Common Approach:

Phase 1: Technology independent optimization
-commit to a particular Boolean network.
-algebraic decomposition used.

Phase 2: AND2/INV decomposition
- commit to a particular decomposition of a general Boolean network using 2-input ANDs (AND2s) and inverters (INVs).

Phase 3: Technology mapping (tree-mapping)

Drawback:

Procedures in each phase are disconnected:

- Phase 1 and Phase 2 make critical decisions without knowing much about constraints and library.
- Phase 3 knows about constraints and library, but solution space is restricted by decisions made earlier.

Lehman-Watanabe Technology Mapping

Incorporate technology independent procedures (Phase 1 and 2) into technology mapping.

Key idea:

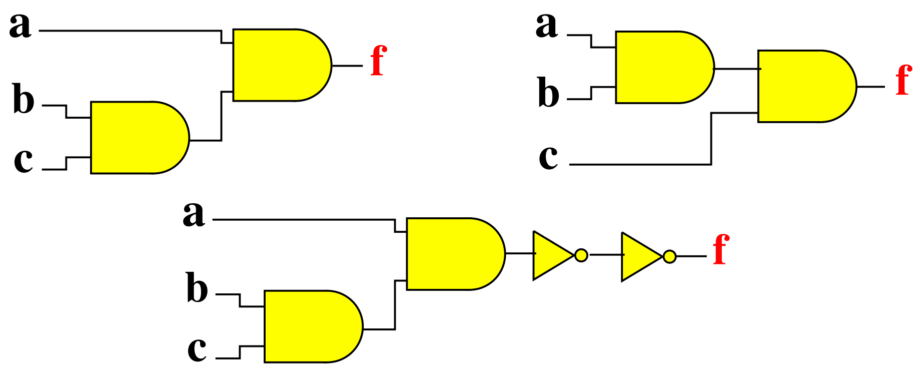
- Efficiently encode a *set* of AND2/INV decompositions into a single structure called a **mapping graph**.
- Apply a modified tree-based technology mapper while **dynamically performing algebraic logic decomposition** on the mapping graph.

Outline

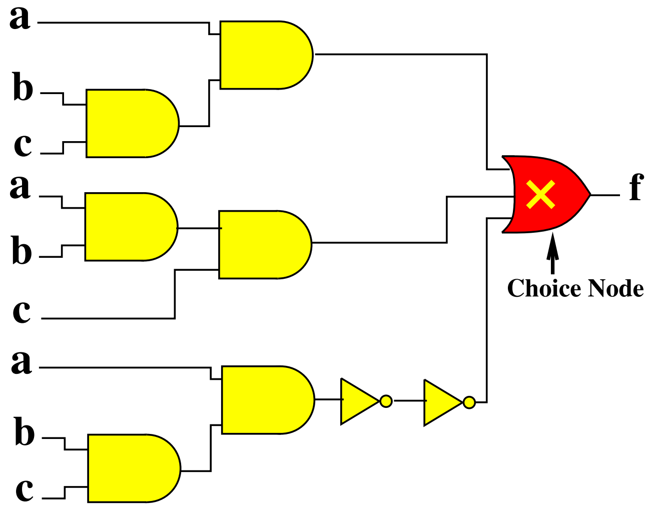
- Mapping graph
  - Encodes a set of AND2/INV decompositions
- Tree-mapping on a mapping graph: **graph-mapping**
- Λ-mapping:
  - without dynamic logic decomposition
  - solution space: Phase 3 + Phase 2
- Δ-mapping:
  - with dynamic logic decomposition
  - solution space: Phase 3 + Phase 2 +
    Algebraic decomposition
    (Phase 1)
- Experimental results

A Set of AND2/INV Decompositions

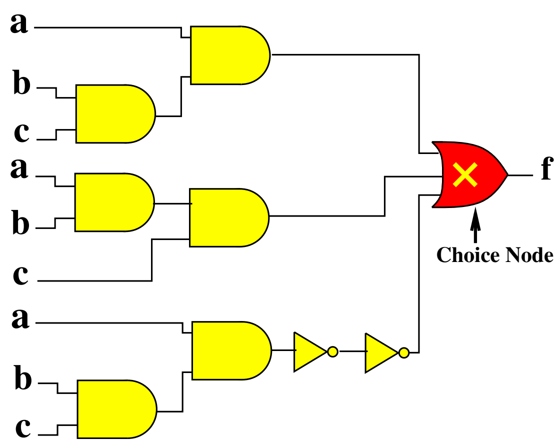
$f = abc$ can be represented in various ways:



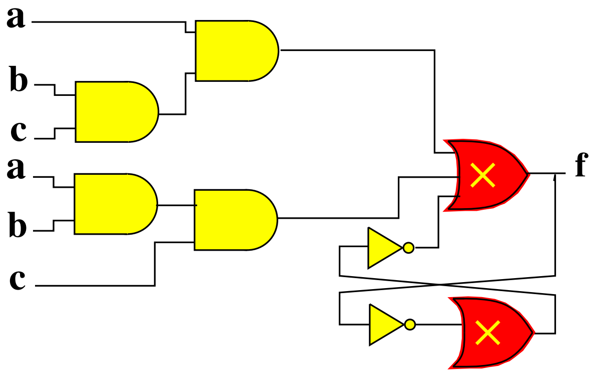
Combine them using a **choice node**:



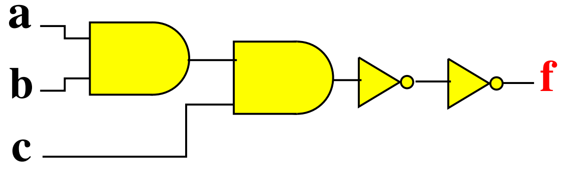
Encoding AND2/INV Decompositions



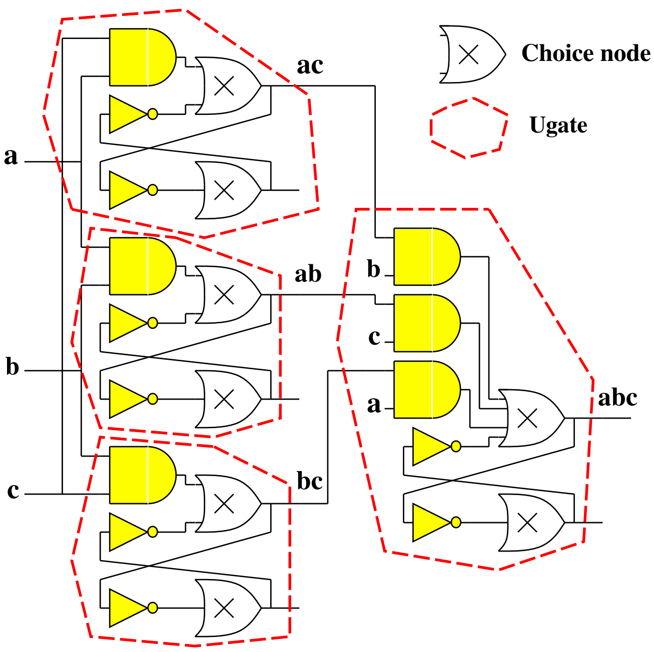
These decompositions can be represented more compactly:



The second representation encodes more decompositions, e.g.



Mapping Graph



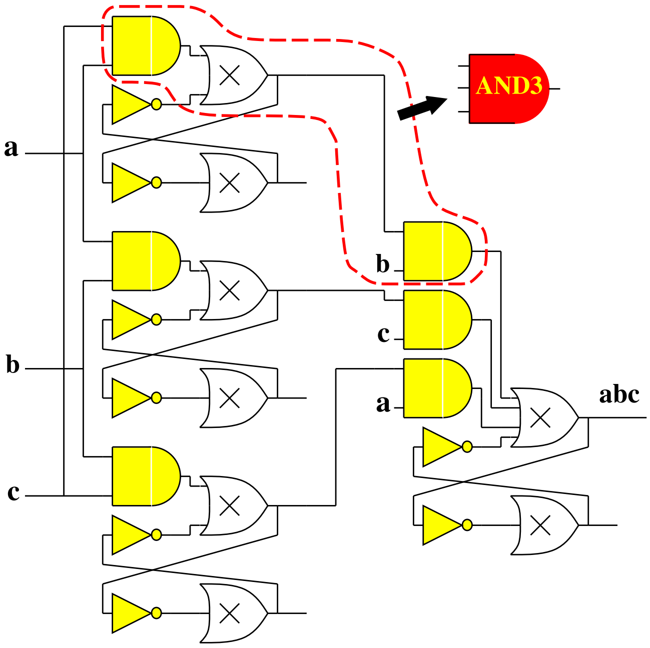
A Boolean network with 4 modifications:

1. **Choice node**: Choices on different decompositions
2. **Cyclic**: Functions written in terms of each other, e.g. inverter chain with an arbitrary length
3. **Reduced**: No two choice nodes with same function. No two AND2s with same fanins.
4. **Ugates**: Efficient implementation

For **cht** benchmark (MCNC91), there are 2.2×10^{93} AND2/INV decompositions. All are encoded only with 400 ugates containing 599 AND2s in total.

Tree-Mapping on a Mapping Graph

Graph-Mapping:



Apply dynamic programming from primary inputs:
(1) find matches at each AND2 and INV, and
(2) retain the cost of a best cover at each node.

- a match may contain choice nodes.
- the cost at a choice node is the minimum of fanin costs.
- fixed-point iteration on each cycle, until costs of all the nodes in the cycle become stable.

Run-time is typically linear in the size of the mapping graph.

Graph-Mapping: Theory

$$\text{graph-mapping}(\mu) = \min_{\theta \in \mu} (\ \text{tree-mapping}(\theta))$$

μ: mapping graph

θ: AND2/INV decomposition encoded in μ

- Graph-mapping finds an optimal tree implementation for each primary output over all the AND2/INV decompositions encoded in μ.
- Graph-mapping is as powerful as applying tree-mapping exhaustively, but is typically exponentially faster.

Λ-Mapping

1. Given a Boolean network η, generate a mapping graph μ: for each node of η,

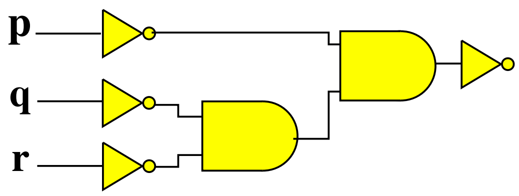
   (1) encode all AND2 decompositions for each product term.

   Ex: $abc \Rightarrow$ Three AND2 decompositions:
   $a(bc)$, $c(ab)$, $b(ca)$.

   (2) encode all AND2/INV decompositions for the sum term.

   Ex: $p+q+r \Rightarrow$ Three AND2/INV decompositions:
   $p + (q + r)$, $r + (p + q)$, $q + (r + p)$.

   $p + (q + r)$ is represented as:

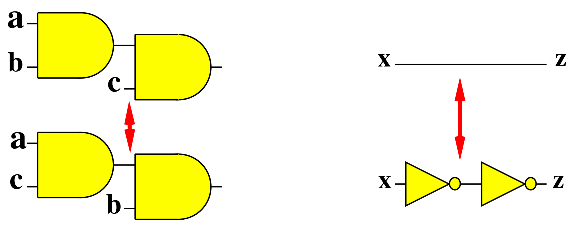
2. Apply graph-mapping on μ.

In practice, η is pre-processed so that each node has at most 10 product terms and each term has at most 10 literals.

Λ-Mapping: Theory

For the mapping graph μ generated for a Boolean network η, let L_η be the set of AND2/INV decompositions encoded in μ.

Let Λ_η be the closure of the set of AND2/INV decompositions of η under the associative and inverter transformations:



Theorem: $\Lambda_\eta \subseteq L_\eta$

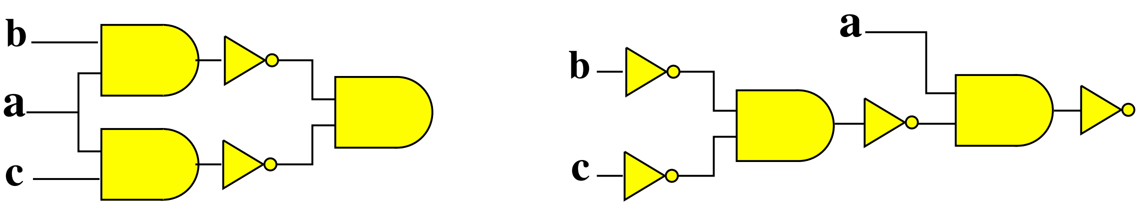
Let θ be an arbitrary AND2/INV decomposition of η.

Theorem: $\theta \in \Lambda_\eta$

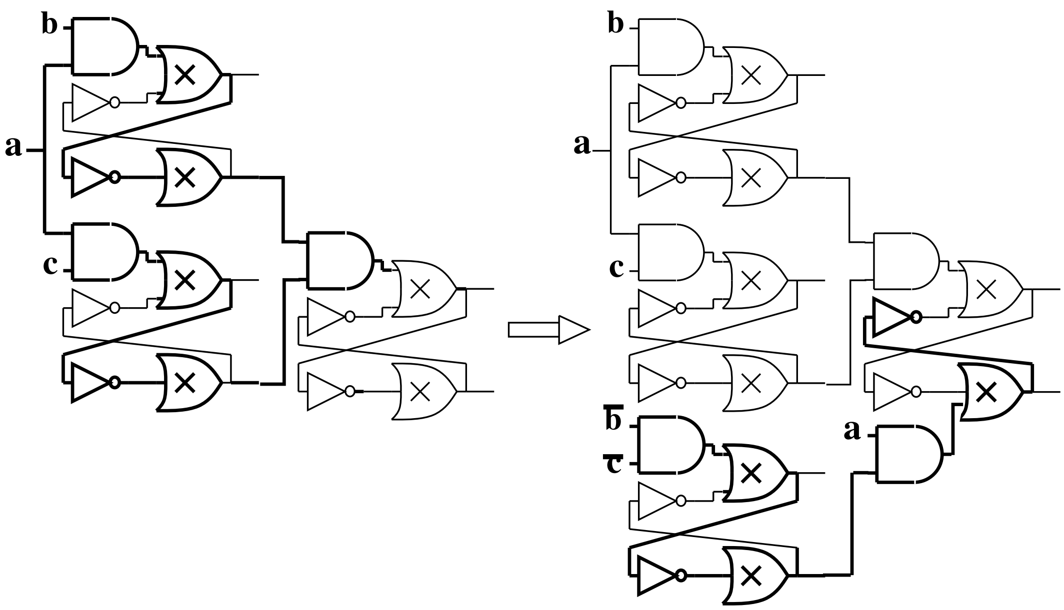
- μ encodes all the AND2/INV decompositions of η.
- Λ-mapping finds at least as good solution as an optimal tree implementation for each primary output over all the AND2/INV decompositions of η.

Dynamic Logic Decomposition

During graph-mapping, dynamically modify the mapping graph: find D-patterns and add F-patterns.



D−pattern: $\overline{ab + ac}$ F−pattern: $\overline{a\,(b + c)}$



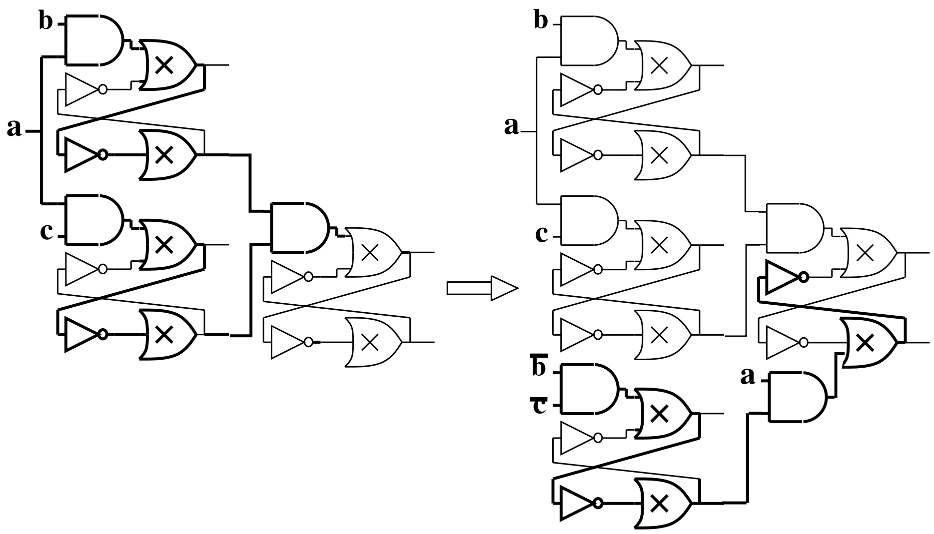
Note: Adding F-pattern may introduce a new D-pattern.

Δ-Mapping

1. Given a Boolean network η, generate a mapping graph μ.

2. Iteratively apply graph-mapping on μ, while performing dynamic logic decomposition until nothing changes in μ.

   - Before find matches at an AND2 in μ, check if D-pattern matches at the AND2. If so, add the corresponding F-pattern.

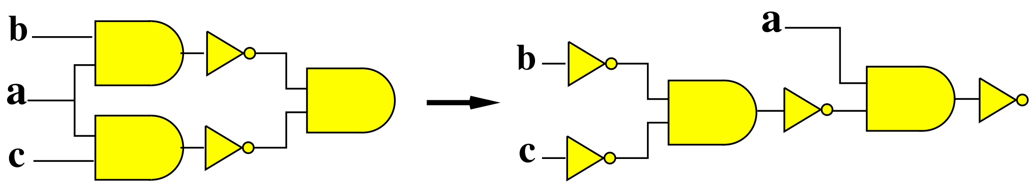


In practice, terminate the procedure when a feasible solution is found.

Δ-Mapping: Theory

For the mapping graph μ generated for a Boolean network η, let D_η be the set of AND2/INV decompositions encoded in the resulting mapping graph.

Let Δ_η be the closure of Λ_η under the distributive transformation:



Theorem: $\Delta_\eta \subseteq D_\eta$

Let $\tilde{\eta}$ be an arbitrary boolean network obtained from η by algebraic decomposition.

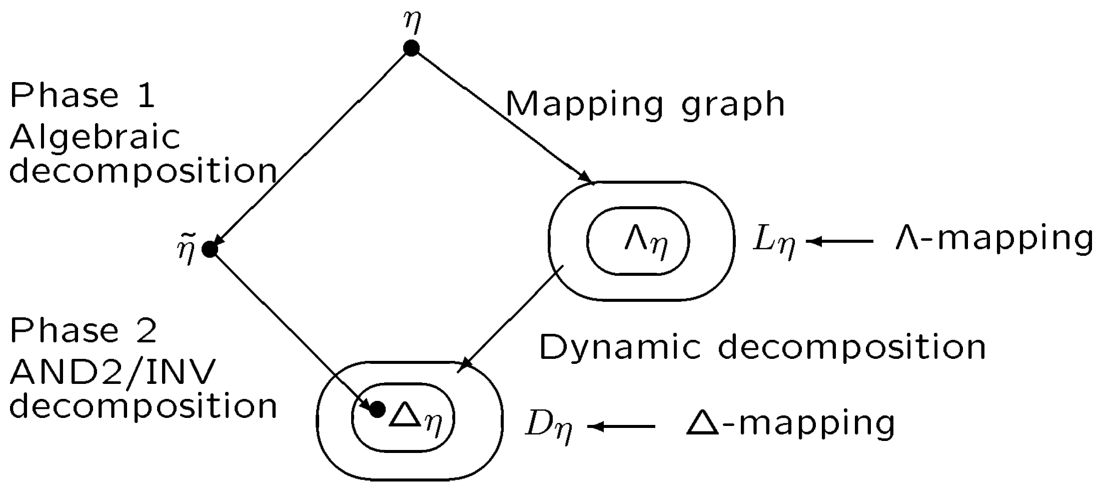
Let θ be an arbitrary AND2/INV decomposition of $\tilde{\eta}$.

Theorem: $\theta \in \Delta_\eta$

- The resulting mapping graph encodes all the AND2/INV decompositions obtained through algebraic decompositions of η.

- Δ-mapping finds at least as good solution as an optimal tree implementation for each primary output over all the AND2/INV decompositions of all the algebraic decompositions of η.

Solution Space captured by Procedures



- Λ-mapping captures all AND2/INV decompositions of η: Phase 2 (subject graph generation) is subsumed.

- Δ-mapping captures all algebraic decompositions: Phase 2 and a key part of Phase 1 are subsumed.

Experiments

Examples: MCNC91 combinational benchmarks

Library: The latest ALPHA microprocessor

Cost: Delay (fastest possible implementation)

1. Apply four different technology independent SIS scripts on the initial Boolean network.
2. Encode each of the four optimized Boolean network into a single mapping graph μ.
3. Apply graph-mapping on μ (Λ-mapping).
4. Collapse the most timing critical region of a mapped network, and encode it in μ.
5. Apply graph-mapping with dynamic logic decomposition (Δ-mapping).

Experimental Results

Name	SIS-1.2		SynFul			
			Λ-mapping		Δ-mapping	
	Delay	Area	Delay	Area	Delay	Area
cc	12.0	167.7	9.5	192.0	8.0	193.3
cm82a	13.2	86.0	10.5	58.3	7.0	78.7
pcle	15.0	266.7	12.0	156.7	10.5	176.7
pm1	13.5	135.0	9.8	135.0	9.2	131.3
x2	17.7	151.0	9.8	109.7	9.8	109.7
cm163a	15.5	130.7	12.0	111.3	9.7	121.0
comp	32.5	401.3	26.0	502.7	21.2	454.3
cu	16.0	145.0	11.8	147.3	9.8	149.0
i1	12.7	183.0	11.5	132.3	9.8	111.0
cm151a	15.0	77.0	10.5	66.3	9.0	88.3
cht	9.2	459.7	9.2	320.7	8.2	323.7
ttt2	21.5	530.7	15.3	576.0	12.7	606.7
cm162a	12.7	164.7	12.0	111.3	9.7	122.7
term1	25.8	612.3	19.5	658.3	15.7	602.0

Delay unit: minimum input-output pin delay over all library gates.

Area unit: minimum gate-area over all library gates.

SynFul's CPU time: ranged from 10 to 300 seconds on VAXstation 6000.

Conclusion

- Logic decomposition during technology mapping
  - Efficiently encode a set of AND2/INV decompositions
  - Dynamically perform logic decomposition
- Two mapping procedures
  - Λ-mapping: optimal over all AND2/INV decompositions
  - Δ-mapping: optimal over all algebraic decompositions
- Implemented and is being used for commercial design projects.
  - Extended for sequential circuits:

    considers all retiming possibilities (implicitly) and algebraic factors across latches.