TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Institut für Computertechnik
Institute of Computer Technology

Logic Synthesis

Digitale Integrierte Schaltungen, 2018 WS

Axel Jantsch

November 6, 2018

Outline

1. Data Structures
2. Two-Level Minimization
3. Multi-Level Minimization
4. Technology Mapping

Logic Synthesis

1. Data Structures
   - Boolean Formulas
   - Representation
   - Conversion
2. Two-Level Minimization
3. Multi-Level Minimization
4. Technology Mapping

Logic Synthesis

1. Data Structures
   - Boolean Formulas
   - Representation
   - Conversion

2. Two-Level Minimization

3. Multi-Level Minimization

4. Technology Mapping

Examples

$$((x_1 \vee (\neg x_2)) \vee ((\neg x_1) \wedge x_3)) \wedge (x_1 \wedge (\neg x_2))$$

Examples

$$((x_1 \vee (\neg x_2)) \vee ((\neg x_1) \wedge x_3)) \wedge (x_1 \wedge (\neg x_2))$$

$$((x_1 \vee \neg x_2) \vee (\neg x_1 \wedge x_3)) \wedge (x_1 \wedge \neg x_2)$$

Examples

$$((x_1 \vee (\neg x_2)) \vee ((\neg x_1) \wedge x_3)) \wedge (x_1 \wedge (\neg x_2))$$

$$((x_1 \vee \neg x_2) \vee (\neg x_1 \wedge x_3)) \wedge (x_1 \wedge \neg x_2)$$

$$(x_1 + \neg x_2 + \neg x_1 x_3) x_1 \neg x_2$$

Examples

$$((x_1 \vee (\neg x_2)) \vee ((\neg x_1) \wedge x_3)) \wedge (x_1 \wedge (\neg x_2))$$

$$((x_1 \vee \neg x_2) \vee (\neg x_1 \wedge x_3)) \wedge (x_1 \wedge \neg x_2)$$

$$(x_1 + \neg x_2 + \neg x_1 x_3) x_1 \neg x_2$$

$$(x_1 \vee \neg x_2 \vee \neg x_1 x_3) x_1 \neg x_2$$

Some Terms

Boolean literal: $\mathbb{B} = \{0, 1\}$, {false,true}

Some Terms

Boolean literal: $\mathbb{B} = \{0, 1\}$, {false,true}

Boolean Variable: $x, y, a, \ldots$

Some Terms

Boolean literal: $\mathbb{B} = \{0, 1\}$, {false,true}

Boolean Variable: $x, y, a, \ldots$

Boolean n-space $\mathbb{B}^n$: n-dimensional Boolean space

Some Terms

Boolean literal:	$\mathbb{B} = \{0, 1\}$, $\{$false,true$\}$
Boolean Variable:	$x, y, a, \ldots$
Boolean n-space $\mathbb{B}^n$:	n-dimensional Boolean space
cube	$a \in \mathbb{B}^n$, "x,y,z", "xyz", "110"

Don't Cares

Completely specified Boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B} = \{0, 1\}$

Don't Cares

Completely specified Boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B} = \{0, 1\}$
Incompletely specified Boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B}_+ = \{0, 1, -\}$

Don't Cares

Completely specified Boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B} = \{0, 1\}$
Incompletely specified Boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B}_+ = \{0, 1, -\}$
Don't care: "-", "X", "x", "2"

Don't Cares

Completely specified Boolean function $f : \mathbb{B}^n \to \mathbb{B} = \{0, 1\}$
Incompletely specified Boolean function $f : \mathbb{B}^n \to \mathbb{B}_+ = \{0, 1, -\}$
Don't care: "-", "X", "x", "2"

$$f(x, y) = \begin{cases} 1 & \text{if } xy = 01 \\ 0 & \text{if } xy = 11 \\ - & \text{otherwise} \end{cases}$$

Onsets and Offsets

For a function $f : \mathbb{B}^n \to \mathbb{B}_+$

onset: $f^{\text{on}} = \{a \in \mathbb{B}^n | f(a) = 1\}$

offset: $f^{\text{off}} = \{a \in \mathbb{B}^n | f(a) = 0\}$

dcset: $f^{\text{dc}} = \{a \in \mathbb{B}^n | f(a) = -\}$

Onsets and Offsets

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	–
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Onsets and Offsets

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	–
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

onset: $f^{\text{on}} = \{000, 011, 100, 101, 110\}$
offset: $f^{\text{off}} = \{001, 111\}$
dcset: $f^{\text{dc}} = \{010\}$

Logic Synthesis

Truth Tables

x	y	$+$
0	0	0
0	1	1
1	0	1
1	1	1

x	y	$\cdot$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$\oplus$, XOR
0	0	0
0	1	1
1	0	1
1	1	0

x	y	$\leftrightarrow$, XNOR
0	0	1
0	1	0
1	0	0
1	1	1

- Canonical
- Effective for few input variables
- **Cubes** are the product terms in the truth table; cubes of the onset of XOR $=$ 01,10

Sum of Products

Sum of Products (SOP), Disjunctive Normal Form (DNF)

$$f = ab\overline{c} + a\overline{b}c + \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- Disjunction of cubes
- The set of cubes of the DNF is a **cover**
- Canonical representation

Sum of Products

Sum of Products (SOP), Disjunctive Normal Form (DNF)

$$f = ab\overline{c} + a\overline{b}c + \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- Disjunction of cubes
- The set of cubes of the DNF is a **cover**
- Canonical representation

Complexity of Operations

Given: m variables,
n terms,

How many terms have to be processed by:

conjunction?

disjunction?

negation?

SAT?

TAUTOLOGY?

Sum of Products - Complexity of Operations

Given:

$$f_1(a, b, c) = ab\overline{c} + a\overline{b}c$$
$$f_2(a, b, c) = \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

Sum of Products - Complexity of Operations

Given:

$$f_1(a,b,c) = ab\overline{c} + a\overline{b}c$$
$$f_2(a,b,c) = \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

Conjunction:

$$\begin{aligned} f_1 \cdot f_2 &= (ab\overline{c} + a\overline{b}c)(\overline{a}bc + \overline{a}\overline{b}\overline{c}) \\ &= ab\overline{c}\,\overline{a}bc + ab\overline{c}\,\overline{a}\overline{b}\overline{c} + a\overline{b}c\overline{a}bc + a\overline{b}c\overline{a}\overline{b}\overline{c} \\ &= 0 + 0 + 0 + 0 \end{aligned}$$

Sum of Products - Complexity of Operations

Given:

$$f_1(a,b,c) = ab\overline{c} + a\overline{b}c$$
$$f_2(a,b,c) = \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

Conjunction:

$$\begin{aligned} f_1 \cdot f_2 &= (ab\overline{c} + a\overline{b}c)(\overline{a}bc + \overline{a}\overline{b}\overline{c}) \\ &= ab\overline{c}\,\overline{a}bc + ab\overline{c}\,\overline{a}\overline{b}\overline{c} + a\overline{b}c\overline{a}bc + a\overline{b}c\overline{a}\overline{b}\overline{c} \\ &= 0 + 0 + 0 + 0 \end{aligned}$$

Conjunction is of quadratic complexity: n^2, for n terms.

Sum of Products - Complexity of Operations

Given:

$$f_1(a,b,c) = ab\overline{c} + a\overline{b}c$$
$$f_2(a,b,c) = \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

Sum of Products - Complexity of Operations

Given:

$$f_1(a,b,c) = ab\overline{c} + a\overline{b}c$$
$$f_2(a,b,c) = \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

Disjunction:

$$f_1 + f_2 = ab\overline{c} + a\overline{b}c + \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

Sum of Products - Complexity of Operations

Given:

$$f_1(a,b,c) = ab\overline{c} + a\overline{b}c$$
$$f_2(a,b,c) = \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

Disjunction:

$$f_1 + f_2 = ab\overline{c} + a\overline{b}c + \overline{a}bc + \overline{a}\overline{b}\overline{c}$$

Disjunction is of constant complexity.

Sum of Products - Complexity of Operations

Given: $f(a, b, c) = ab\overline{c} + a\overline{b}c$

Sum of Products - Complexity of Operations

Given: $f(a,b,c) = ab\overline{c} + a\overline{b}c$

Complement:

$$\begin{aligned}
\overline{f} &= \overline{(ab\overline{c} + a\overline{b}c)} \\
&= \overline{(ab\overline{c})}\,\overline{(a\overline{b}c)} \\
&= (\overline{a} + \overline{b} + c)(\overline{a} + b + \overline{c}) \\
&= \overline{a}\overline{a} + \overline{a}b + \overline{a}\overline{c} + \overline{b}\overline{a} + \overline{b}b + \overline{b}\overline{c} + c\overline{a} + cb + c\overline{c}
\end{aligned}$$

Sum of Products - Complexity of Operations

Given: $f(a,b,c) = ab\overline{c} + a\overline{b}c$

Complement:

$$\begin{aligned}
\overline{f} &= \overline{(ab\overline{c} + a\overline{b}c)} \\
&= \overline{(ab\overline{c})}\,\overline{(a\overline{b}c)} \\
&= (\overline{a} + \overline{b} + c)(\overline{a} + b + \overline{c}) \\
&= \overline{a}\overline{a} + \overline{a}b + \overline{a}\overline{c} + \overline{b}\overline{a} + \overline{b}b + \overline{b}\overline{c} + c\overline{a} + cb + c\overline{c}
\end{aligned}$$

Complementing function

$$f = x_{1,1}x_{1,2}\cdots x_{1,m} + \cdots + x_{n,1}x_{n,2}\cdots x_{n,m}$$

will result in m^n product terms in the SOP representation.
Complement is of exponential complexity.

SAT - Satisfiability of a Function

A function $f(x_1, \cdots, x_m)$ is satisfiable if there exists a consistent variable assignment, such that $f = 1$

SAT - Satisfiability of a Function

A function $f(x_1, \cdots, x_m)$ is satisfiable if there exists a consistent variable assignment, such that $f = 1$

Examples:

SAT - Satisfiability of a Function

A function $f(x_1, \cdots, x_m)$ is satisfiable if there exists a consistent variable assignment, such that $f = 1$

Examples:

$$f_1(x, y) = \overline{x}y \quad \cdots \quad \text{satisfiable since } f_1(0, 1) = 1$$

SAT - Satisfiability of a Function

A function $f(x_1, \cdots, x_m)$ is satisfiable if there exists a consistent variable assignment, such that $f = 1$

Examples:

$$\begin{aligned} f_1(x, y) = \overline{x}y \quad & \cdots \quad \text{satisfiable since } f_1(0,1) = 1 \\ f_2(x) = x\overline{x} \quad & \cdots \quad \text{is not satisfiable.} \end{aligned}$$

SAT - Satisfiability of a Function

A function $f(x_1, \cdots, x_m)$ is satisfiable if there exists a consistent variable assignment, such that $f = 1$

Examples:

$$f_1(x, y) = \overline{x}y \quad \cdots \quad \text{satisfiable since } f_1(0, 1) = 1$$
$$f_2(x) = x\overline{x} \quad \cdots \quad \text{is not satisfiable.}$$

SAT is NP complete.

(Cook-Levin Theorem from 1971/73.)

SAT - Satisfiability of a Function

A function $f(x_1, \cdots, x_m)$ is satisfiable if there exists a consistent variable assignment, such that $f = 1$

Examples:

$$\begin{aligned} f_1(x, y) &= \overline{x}y \quad \cdots \quad \text{satisfiable since } f_1(0,1) = 1 \\ f_2(x) &= x\overline{x} \quad \cdots \quad \text{is not satisfiable.} \end{aligned}$$

SAT is NP complete.

(Cook-Levin Theorem from 1971/73.)

For SOP formulas SAT can be determined in constant time.

Tautology Checking

A function $f(x_1, \cdots, x_m)$ is a tautology if $f = 1$ for every consistent variable assignment.

Tautology Checking

A function $f(x_1, \cdots, x_m)$ is a tautology if $f = 1$ for every consistent variable assignment.

Examples:

Tautology Checking

A function $f(x_1, \cdots, x_m)$ is a tautology if $f = 1$ for every consistent variable assignment.

Examples:

$$f_1(x, y) = \overline{x}y \quad \cdots \quad \text{is no tautology since } f_1(1, 1) = 0$$

Tautology Checking

A function $f(x_1, \cdots, x_m)$ is a tautology if $f = 1$ for every consistent variable assignment.

Examples:

$$f_1(x, y) = \overline{x}y \quad \cdots \quad \text{is no tautology since } f_1(1, 1) = 0$$
$$f_2(x) = x + \overline{x} \quad \cdots \quad \text{is a tautology.}$$

Tautology Checking

A function $f(x_1, \cdots, x_m)$ is a tautology if $f = 1$ for every consistent variable assignment.

Examples:

$$f_1(x, y) = \overline{x}y \quad \cdots \quad \text{is no tautology since } f_1(1,1) = 0$$

$$f_2(x) = x + \overline{x} \quad \cdots \quad \text{is a tautology.}$$

Function f is a tautology iff $\overline{f}$ is not satisfiable.

Tautology Checking

A function $f(x_1, \cdots, x_m)$ is a tautology if $f = 1$ for every consistent variable assignment.

Examples:

$$
\begin{aligned}
f_1(x,y) = \overline{x}y \quad & \cdots \quad \text{is no tautology since } f_1(1,1) = 0 \\
f_2(x) = x + \overline{x} \quad & \cdots \quad \text{is a tautology.}
\end{aligned}
$$

Function f is a tautology iff $\overline{f}$ is not satisfiable.

TAUTOLOGY is NP complete.

Tautology Checking

A function $f(x_1, \cdots, x_m)$ is a tautology if $f = 1$ for every consistent variable assignment.

Examples:

$$
\begin{aligned}
f_1(x, y) = \overline{x}y \quad & \cdots \quad \text{is no tautology since } f_1(1,1) = 0 \\
f_2(x) = x + \overline{x} \quad & \cdots \quad \text{is a tautology.}
\end{aligned}
$$

Function f is a tautology iff $\overline{f}$ is not satisfiable.

TAUTOLOGY is NP complete.

For SOP formulas we need to test 2^m assignments; hence, TAUTOLOGY is determined in exponential time.

Sum of Products - Complexity of Operations

Operation	Complexity
Conjunction	Quadratic: n^2
Disjunction	Constant
Complement	**Exponential**: m^n
SAT	Constant
TAUTOLOGY	**Exponential**: 2^m

Product of Sums

Product of Sums (POS), Conjunctive Normal Form (CNF, KNF)

$$f = (a + b + \overline{c})(a + \overline{b} + c)(\overline{a} + b + c)(\overline{a} + \overline{b} + \overline{c})$$

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- Conjunctions of disjunctions
- Canonical representation
- POS is the dual of SOP
- Less widely used than SOP

Product of Sums

Product of Sums (POS), Conjunctive Normal Form (CNF, KNF)

$$f = (a + b + \overline{c})(a + \overline{b} + c)(\overline{a} + b + c)(\overline{a} + \overline{b} + \overline{c})$$

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- Conjunctions of disjunctions
- Canonical representation
- POS is the dual of SOP
- Less widely used than SOP

Products of Sums - Complexity of Operations

Given:

$$f_1(a, b, c) = (a + b + \overline{c})(a + \overline{b} + c)$$
$$f_2(a, b, c) = (\overline{a} + b + c)(\overline{a} + \overline{b} + \overline{c})$$

Products of Sums - Complexity of Operations

Given:

$$f_1(a,b,c) = (a+b+\overline{c})(a+\overline{b}+c)$$
$$f_2(a,b,c) = (\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c})$$

Conjunction:

$$f_1 \cdot f_2 = (a+b+\overline{c})(a+\overline{b}+c)(\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c})$$

Products of Sums - Complexity of Operations

Given:

$$f_1(a, b, c) = (a + b + \overline{c})(a + \overline{b} + c)$$
$$f_2(a, b, c) = (\overline{a} + b + c)(\overline{a} + \overline{b} + \overline{c})$$

Conjunction:

$$f_1 \cdot f_2 = (a + b + \overline{c})(a + \overline{b} + c)(\overline{a} + b + c)(\overline{a} + \overline{b} + \overline{c})$$

Conjunction is of constant complexity.

Products of Sums - Complexity of Operations

Given:

$$f_1(a, b, c) = (a + b + \overline{c})(a + \overline{b} + c)$$
$$f_2(a, b, c) = (\overline{a} + b + c)(\overline{a} + \overline{b} + \overline{c})$$

Products of Sums - Complexity of Operations

Given:

$$f_1(a,b,c) = (a+b+\overline{c})(a+\overline{b}+c)$$
$$f_2(a,b,c) = (\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c})$$

Disjunction:

$$f_1 + f_2 = (a+b+\overline{c})(a+\overline{b}+c) + (\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c})$$

Products of Sums - Complexity of Operations

Given:

$$f_1(a,b,c) = (a+b+\overline{c})(a+\overline{b}+c)$$
$$f_2(a,b,c) = (\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c})$$

Disjunction:

$$\begin{aligned} f_1 + f_2 &= (a+b+\overline{c})(a+\overline{b}+c) + (\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c}) \\ &= x_1x_2 + x_3x_4 \\ &= (x_1+x_3)(x_2+x_3)(x_1+x_4)(x_2+x_4) \qquad \text{(distributive law)} \end{aligned}$$

Products of Sums - Complexity of Operations

Given:

$$f_1(a,b,c) = (a+b+\overline{c})(a+\overline{b}+c)$$
$$f_2(a,b,c) = (\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c})$$

Disjunction:

$$\begin{aligned}
f_1 + f_2 &= (a+b+\overline{c})(a+\overline{b}+c) + (\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c}) \\
&= x_1x_2 + x_3x_4 \\
&= (x_1+x_3)(x_2+x_3)(x_1+x_4)(x_2+x_4) \qquad \text{(distributive law)} \\
&= ((a+b+\overline{c})+(\overline{a}+b+c)) \\
&\quad ((a+\overline{b}+c)+(\overline{a}+b+c)) \\
&\quad ((a+b+\overline{c})+(\overline{a}+\overline{b}+\overline{c})) \\
&\quad ((a+\overline{b}+c)+(\overline{a}+\overline{b}+\overline{c})
\end{aligned}$$

Products of Sums - Complexity of Operations

Given:

$$f_1(a,b,c) = (a+b+\overline{c})(a+\overline{b}+c)$$
$$f_2(a,b,c) = (\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c})$$

Disjunction:

$$\begin{aligned}
f_1 + f_2 &= (a+b+\overline{c})(a+\overline{b}+c)+(\overline{a}+b+c)(\overline{a}+\overline{b}+\overline{c}) \\
&= x_1x_2 + x_3x_4 \\
&= (x_1+x_3)(x_2+x_3)(x_1+x_4)(x_2+x_4) \qquad \text{(distributive law)} \\
&= ((a+b+\overline{c})+(\overline{a}+b+c)) \\
&\quad ((a+\overline{b}+c)+(\overline{a}+b+c)) \\
&\quad ((a+b+\overline{c})+(\overline{a}+\overline{b}+\overline{c})) \\
&\quad ((a+\overline{b}+c)+(\overline{a}+\overline{b}+\overline{c})
\end{aligned}$$

Disjunction is of quadratic complexity: n^2 for n terms.

Product of Sums - Complexity of Operations

Given: $f(a,b,c) = (a+b+\overline{c})(a+\overline{b}+c)$

Product of Sums - Complexity of Operations

Given: $f(a,b,c) = (a+b+\overline{c})(a+\overline{b}+c)$

Complement:

$$
\begin{aligned}
\overline{f} &= \overline{(a+b+\overline{c})} + \overline{(a+\overline{b}+c)} \\
&= (\overline{a}\,\overline{b}c) + (\overline{a}b\overline{c}) \\
&= (\overline{a}+\overline{a})(\overline{a}+b)(\overline{a}+\overline{c})(\overline{b}+\overline{a})(\overline{b}+b)(\overline{b}+\overline{c}) \\
&\quad (c+\overline{a})(c+b)(c+\overline{c})
\end{aligned}
$$

Product of Sums - Complexity of Operations

Given: $f(a,b,c) = (a+b+\overline{c})(a+\overline{b}+c)$

Complement:

$$\begin{aligned}
\overline{f} &= \overline{(a+b+\overline{c})} + \overline{(a+\overline{b}+c)} \\
&= (\overline{a}\overline{b}c) + (\overline{a}b\overline{c}) \\
&= (\overline{a}+\overline{a})(\overline{a}+b)(\overline{a}+\overline{c})(\overline{b}+\overline{a})(\overline{b}+b)(\overline{b}+\overline{c}) \\
&\quad (c+\overline{a})(c+b)(c+\overline{c})
\end{aligned}$$

Complementing function

$$f = (x_{1,1} + x_{1,2} + \cdots + x_{1,m}) \cdots (x_{n,1} + x_{n,2} + \cdots + x_{n,m})$$

will result in m^n sum terms in the POS representation.
Complement is of exponential complexity.

Product of Sums - Complexity of Operations

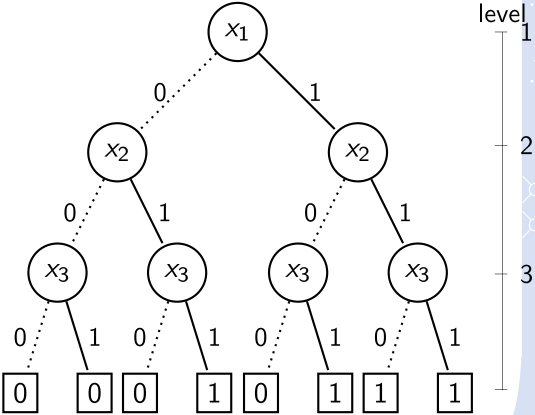
Operation	SOP	POS
Conjunction	Quadratic: n^2	Constant
Disjunction	Constant	Quadratic: n^2
Complement	**Exponential**: m^n	**Exponential**: m^n
SAT	Constant	**Exponential**: 2^m
TAUTOLOGY	**Exponential**: 2^m	Constant

Binary Decision Diagram

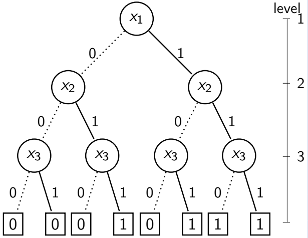
x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Binary Decision Diagram

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

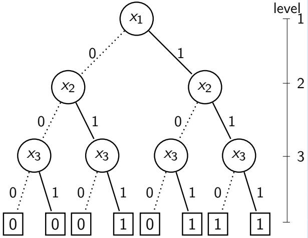


Binary Decision Diagram



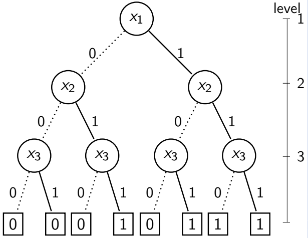
Binary Decision Diagram

- An n-level binary tree represents an n-variable Boolean function.



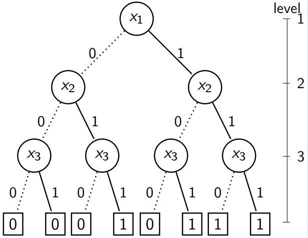
Binary Decision Diagram

- An n-level binary tree represents an n-variable Boolean function.
- A **terminal node** v has attribute value$(v) \in \{0, 1\}$.



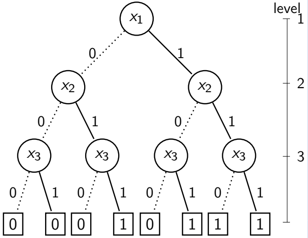
Binary Decision Diagram

- An n-level binary tree represents an n-variable Boolean function.
- A **terminal node** v has attribute value$(v) \in \{0, 1\}$.
- A **non-terminal node** v has an index$(v) \in \{1..n\}$, and two children: the 0-child (else(v)) and the 1-child (then(v)).



Binary Decision Diagram

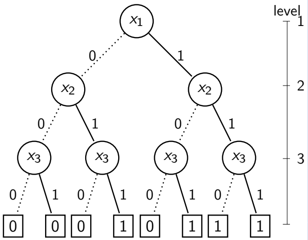
- An n-level binary tree represents an n-variable Boolean function.
- A **terminal node** v has attribute value$(v) \in \{0, 1\}$.
- A **non-terminal node** v has an index$(v) \in \{1..n\}$, and two children: the 0-child (else(v)) and the 1-child (then(v)).



- The variable $x_{\text{index}(v)}$ is the decision variable for node v.

Binary Decision Diagram

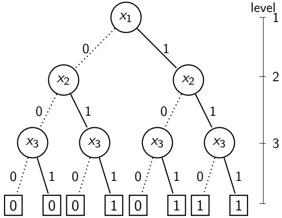
- An n-level binary tree represents an n-variable Boolean function.
- A **terminal node** v has attribute $\text{value}(v) \in \{0, 1\}$.
- A **non-terminal node** v has an $\text{index}(v) \in \{1..n\}$, and two children: the 0-child ($\text{else}(v)$) and the 1-child ($\text{then}(v)$).



- The variable $x_{\text{index}(v)}$ is the decision variable for node v.
- Every node v corresponds to a Boolean function f_v recursively defined:
  1. If v is a terminal node, $f_v = \text{value}(v)$;
  2. If v is not a terminal node:

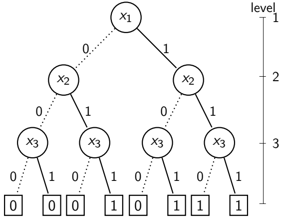
$$f_v(x_1, \ldots, x_n) = \overline{x_{\text{index}(v)}} f_{\text{else}(v)}(x_1, \ldots, x_n) + x_{\text{index}(v)} f_{\text{then}(v)}(x_1, \ldots, x_n)$$

Ordered Binary Decision Diagram



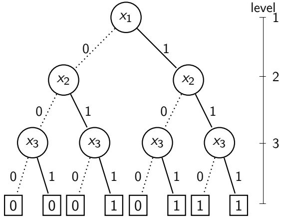
Ordered Binary Decision Diagram

- A BDD is **ordered** (it is an OBDD) if the nodes on every path from the root to a terminal node follow the same variable ordering.



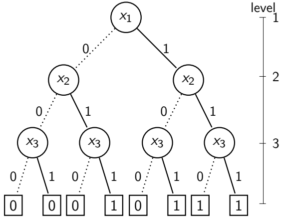
Ordered Binary Decision Diagram

- A BDD is **ordered** (it is an OBDD) if the nodes on every path from the root to a terminal node follow the same variable ordering.
- Each OBDD represents a Boolean function.



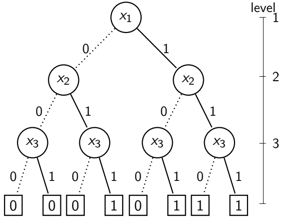
Ordered Binary Decision Diagram

- A BDD is **ordered** (it is an OBDD) if the nodes on every path from the root to a terminal node follow the same variable ordering.
- Each OBDD represents a Boolean function.
- Each Boolean function is represented by an OBDD.



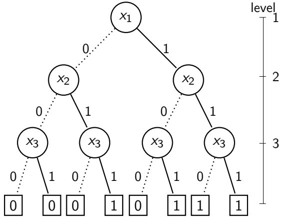
Ordered Binary Decision Diagram

- A BDD is **ordered** (it is an OBDD) if the nodes on every path from the root to a terminal node follow the same variable ordering.
- Each OBDD represents a Boolean function.
- Each Boolean function is represented by an OBDD.
- Two OBDDs are isomorphic if they represent the same Boolean function.

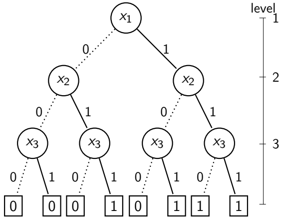


Ordered Binary Decision Diagram

- A BDD is **ordered** (it is an OBDD) if the nodes on every path from the root to a terminal node follow the same variable ordering.
- Each OBDD represents a Boolean function.
- Each Boolean function is represented by an OBDD.
- Two OBDDs are isomorphic if they represent the same Boolean function.
- The isomorphism of two OBDDs can be checked in linear time with the size of the tree.

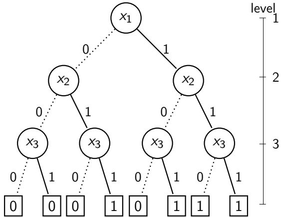


Reduced Ordered Binary Decision Diagram



Reduced Ordered Binary Decision Diagram

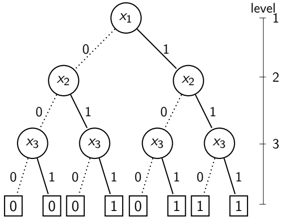
A **reduced OBDD (ROBDD)** is constructed from an OBDD as follows:



Reduced Ordered Binary Decision Diagram

A **reduced OBDD (ROBDD)** is constructed from an OBDD as follows:

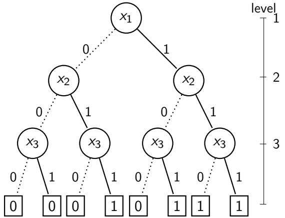
1. Two terminal nodes with the same value attribute are merged.



Reduced Ordered Binary Decision Diagram

A **reduced OBDD (ROBDD)** is constructed from an OBDD as follows:

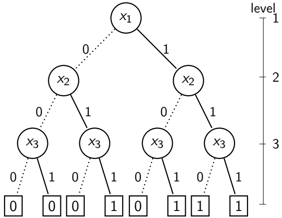
1. Two terminal nodes with the same value attribute are merged.
2. Two non-terminal nodes u and v with the same decision variable, the same 0-child ($\text{else}(v) = \text{else}(u)$) and he same 1-child ($\text{then}(v) = \text{then}(u)$) are merged.



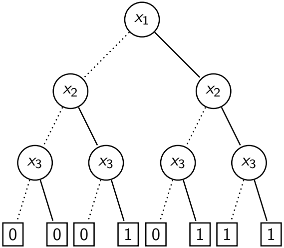
Reduced Ordered Binary Decision Diagram

A **reduced OBDD (ROBDD)** is constructed from an OBDD as follows:

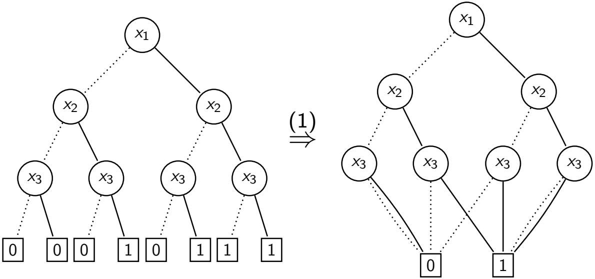
1. Two terminal nodes with the same value attribute are merged.
2. Two non-terminal nodes u and v with the same decision variable, the same 0-child ($\text{else}(v) = \text{else}(u)$) and he same 1-child ($\text{then}(v) = \text{then}(u)$) are merged.
3. A non-terminal node v with $\text{then}(v) = \text{else}(u)$ is removed and its incident edges are redirected to its child node.



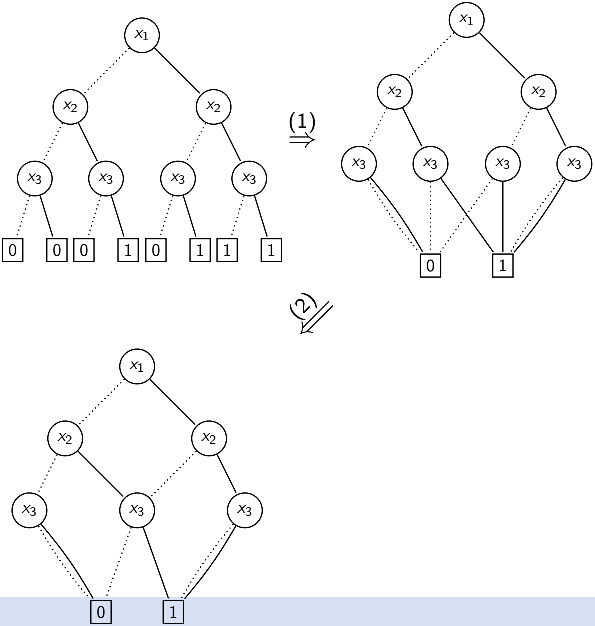
Reduced Ordered Binary Decision Diagram



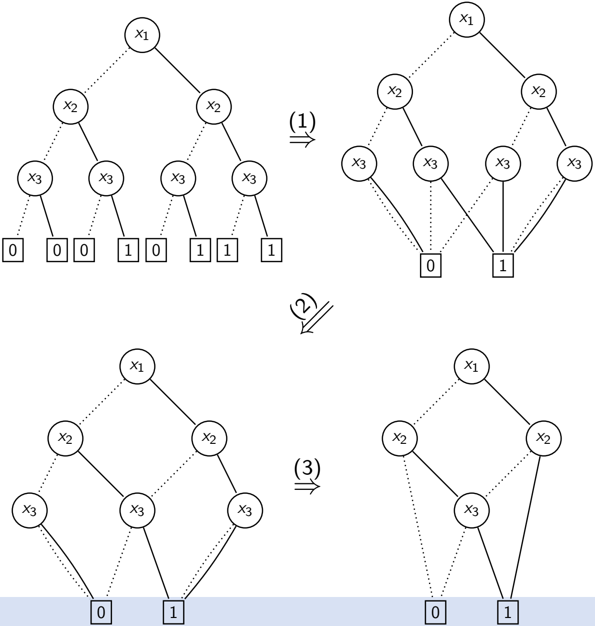
Reduced Ordered Binary Decision Diagram



Reduced Ordered Binary Decision Diagram



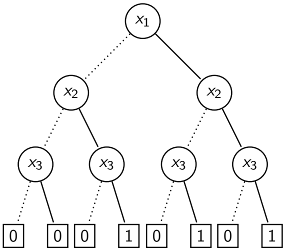
Reduced Ordered Binary Decision Diagram



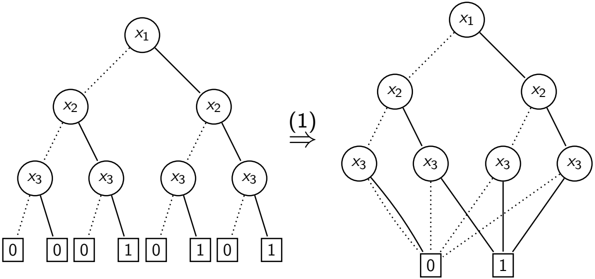
Reduced Ordered Binary Decision Diagram

- ROBBD has smallest number of nodes under a given variable ordering.
- ROBBD is a canonical representation of a Boolean function for a given variable ordering.

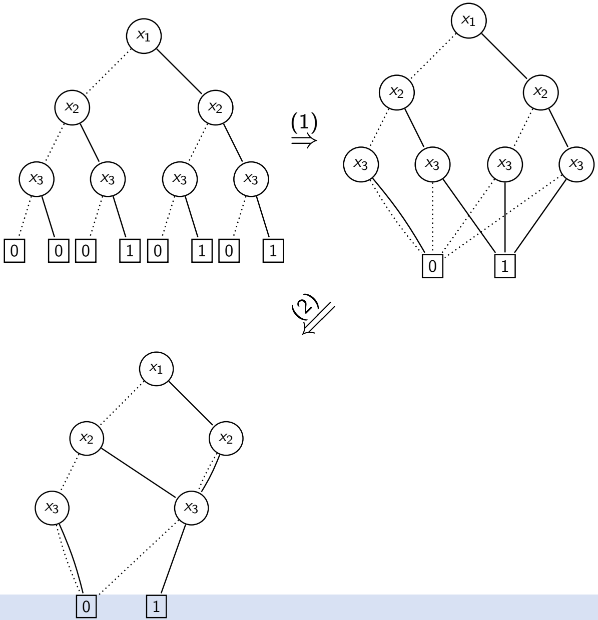
ROBDD Generation Example



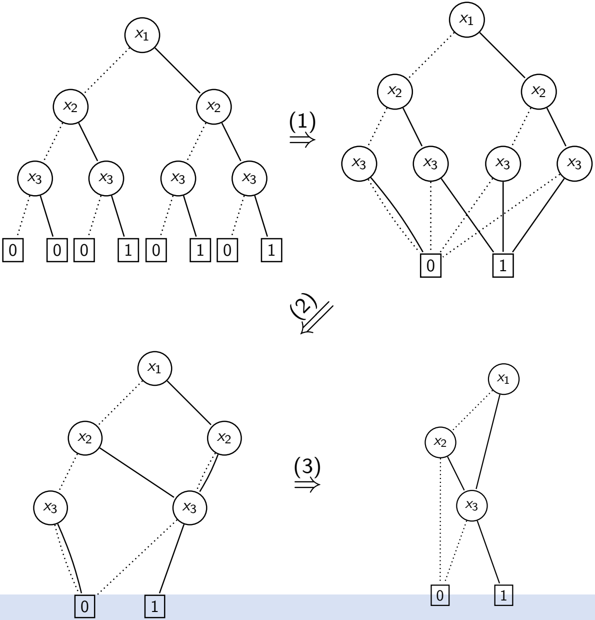
ROBDD Generation Example



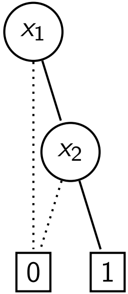
ROBDD Generation Example



ROBDD Generation Example



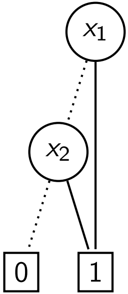
ROBDDs represent onsets and offsets: AND



x_1	x_2	$\cdot$
0	0	0
0	1	0
1	0	0
1	1	1

onset: $\{x_1 x_2\}$
offset: $\{\overline{x_1}, x_1\overline{x_2}\}$

ROBDDs represent onsets and offsets: OR

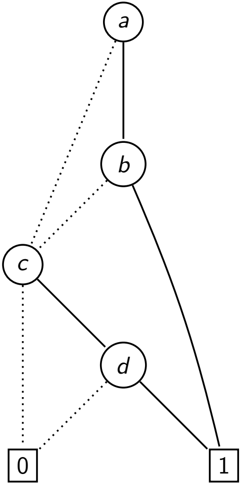


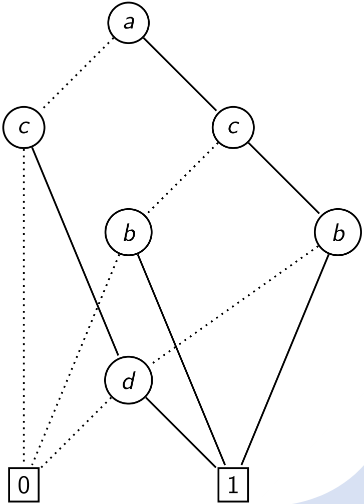
x_1	x_2	$+$
0	0	0
0	1	1
1	0	1
1	1	1

onset: $\{x_1, \overline{x_1}x_2\}$
offset: $\{\overline{x_1 x_2}\}$

ROBDD and Variable Ordering

Function: $f = ab + cd$





ROBDD and Variable Ordering

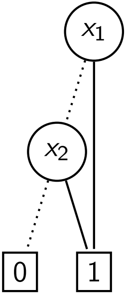
- Size of the ROBDD varies strongly with the ordering of variables.
- Finding the optimal variable ordering is of exponential complexity.
- There are good heuristics for finding a good ordering.
- There exist functions with ROBDD size growing exponential with the size of the Boolean formula **for all orderings**, e.g. multiplication.

Computational Efficiency of ROBBDs

Complement is done by complementing the terminal nodes.

Computational Efficiency of ROBBDs

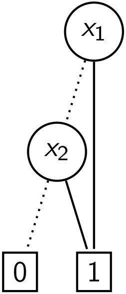
Complement is done by complementing the terminal nodes.



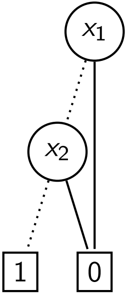
x_1	x_2	$+$
0	0	0
0	1	1
1	0	1
1	1	1

Computational Efficiency of ROBBDs

Complement is done by complementing the terminal nodes.



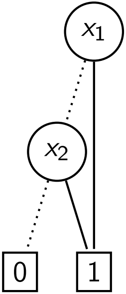
x_1	x_2	$+$
0	0	0
0	1	1
1	0	1
1	1	1



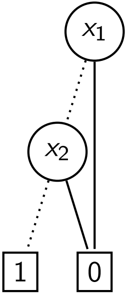
x_1	x_2	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Computational Efficiency of ROBBDs

Complement is done by complementing the terminal nodes.



x_1	x_2	$+$
0	0	0
0	1	1
1	0	1
1	1	1



x_1	x_2	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Complement is of constant complexity.

Computational Efficiency of ROBBDs

Binary operations are done based on the Shannon Expansion.

$$
\begin{aligned}
f(x_1, \ldots, x_i, \ldots, x_n) &= (x_i \cdot f_{x_i}) + (\overline{x_i} \cdot f_{\overline{x_i}}) \\
&= (x_i \cdot f(x_1, \ldots, 1, \ldots, x_n)) \\
&\quad + \ (\overline{x_i} \cdot f(x_1, \ldots, 0, \ldots, x_n))
\end{aligned}
$$

Computational Efficiency of ROBBDs

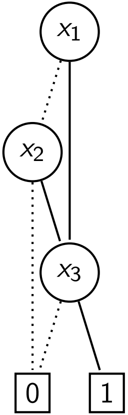
Binary operations are done based on the Shannon Expansion.

$$
\begin{aligned}
f(x_1,\ldots,x_i,\ldots,x_n) &= (x_i \cdot f_{x_i}) + (\overline{x_i} \cdot f_{\overline{x_i}}) \\
&= (x_i \cdot f(x_1,\ldots,1,\ldots,x_n)) \\
&\quad + \ (\overline{x_i} \cdot f(x_1,\ldots,0,\ldots,x_n))
\end{aligned}
$$

$$
f\,\langle \mathrm{op}\rangle g = (x_i \cdot (f_{x_i}\langle \mathrm{op}\rangle g_{x_i})) + (\overline{x_i} \cdot (f_{\overline{x_i}}\langle \mathrm{op}\rangle g_{\overline{x_i}}))
$$

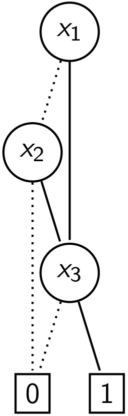
Shannon Expansion with ROBBDs

$f = \overline{x_1}x_2x_3 + x_1x_3$

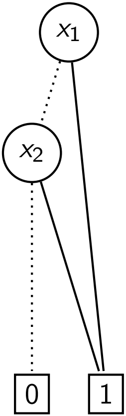


Shannon Expansion with ROBBDs

$f = \overline{x_1} x_2 x_3 + x_1 x_3$

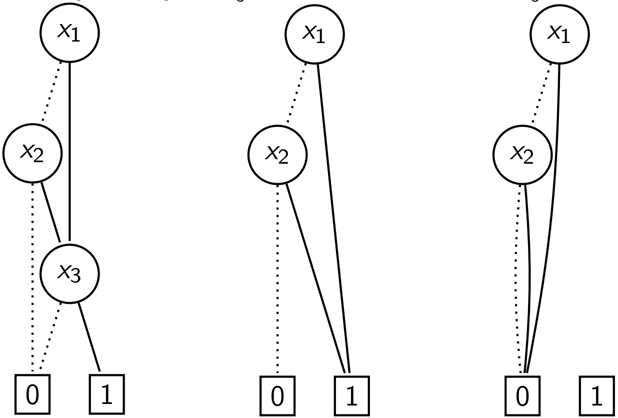


$f_{x_3} = \overline{x_1} x_2 + x_1$



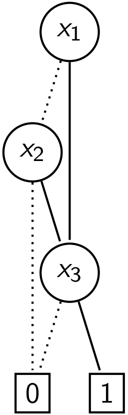
Shannon Expansion with ROBBDs

$f = \overline{x_1}x_2x_3 + x_1x_3$ $\quad$ $f_{x_3} = \overline{x_1}x_2 + x_1$ $\quad$ $f_{\overline{x_3}} = 0$



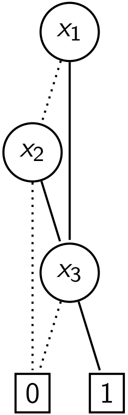
Shannon Expansion with ROBBDs

$f = \overline{x_1} x_2 x_3 + x_1 x_3$

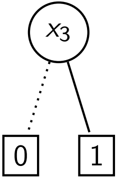


Shannon Expansion with ROBBDs

$f = \overline{x_1}x_2x_3 + x_1x_3$

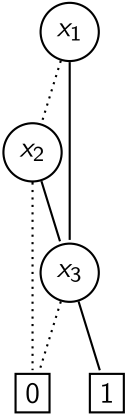


$f_{x_1} = x_3$

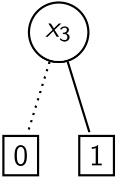


Shannon Expansion with ROBBDs

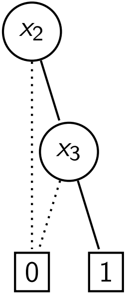
$f = \overline{x_1}x_2x_3 + x_1x_3$



$f_{x_1} = x_3$



$f_{\overline{x_1}} = x_2x_3$



Binary Operation by Way of Shannon Expansion

$$\begin{aligned} f(a,b) &= a \cdot b \\ g(a,b) &= a + b \end{aligned}$$

Binary Operation by Way of Shannon Expansion

$$
\begin{aligned}
f(a,b) &= a \cdot b \\
g(a,b) &= a + b \\
f + g &= (a \cdot b) + (a + b) = (a \cdot b) + a + b = a + b
\end{aligned}
$$

Binary Operation by Way of Shannon Expansion

$$
\begin{aligned}
f(a,b) &= a \cdot b \\
g(a,b) &= a + b \\
f + g &= (a \cdot b) + (a + b) = (a \cdot b) + a + b = a + b \\
f + g &= (a \cdot (f_a + g_a)) + (\overline{a} \cdot (f_{\overline{a}} + g_{\overline{a}}))
\end{aligned}
$$

Binary Operation by Way of Shannon Expansion

$$
\begin{aligned}
f(a,b) &= a \cdot b \\
g(a,b) &= a + b \\
f + g &= (a \cdot b) + (a + b) = (a \cdot b) + a + b = a + b \\
f + g &= (a \cdot (f_a + g_a)) + (\overline{a} \cdot (f_{\overline{a}} + g_{\overline{a}})) \\
&= (a \cdot ((b \cdot (f_{a,b} + g_{a,b})) + (\overline{b} \cdot (f_{a,\overline{b}} + g_{a,\overline{b}})))) \\
&\quad + (\overline{a} \cdot ((b \cdot (f_{\overline{a},b} + g_{\overline{a},b})) + (\overline{b} \cdot (f_{\overline{a},\overline{b}} + g_{\overline{a},\overline{b}}))))
\end{aligned}
$$

Binary Operation by Way of Shannon Expansion

$$
\begin{aligned}
f(a,b) &= a \cdot b \\
g(a,b) &= a + b \\
f + g &= (a \cdot b) + (a + b) = (a \cdot b) + a + b = a + b \\
f + g &= (a \cdot (f_a + g_a)) + (\overline{a} \cdot (f_{\overline{a}} + g_{\overline{a}})) \\
&= (a \cdot ((b \cdot (f_{a,b} + g_{a,b})) + (\overline{b} \cdot (f_{a,\overline{b}} + g_{a,\overline{b}})))) \\
&\quad + (\overline{a} \cdot ((b \cdot (f_{\overline{a},b} + g_{\overline{a},b})) + (\overline{b} \cdot (f_{\overline{a},\overline{b}} + g_{\overline{a},\overline{b}})))) \\
&= (a \cdot ((b \cdot (1 + 1)) + (\overline{b} \cdot (0 + 1)))) \\
&\quad + (\overline{a} \cdot ((b \cdot (0 + 1)) + (\overline{b} \cdot (0 + 0))))
\end{aligned}
$$

Binary Operation by Way of Shannon Expansion

$$
\begin{aligned}
f(a,b) &= a \cdot b \\
g(a,b) &= a + b \\
f+g &= (a \cdot b) + (a+b) = (a \cdot b) + a + b = a + b \\
f+g &= (a \cdot (f_a + g_a)) + (\overline{a} \cdot (f_{\overline{a}} + g_{\overline{a}})) \\
&= (a \cdot ((b \cdot (f_{a,b} + g_{a,b})) + (\overline{b} \cdot (f_{a,\overline{b}} + g_{a,\overline{b}})))) \\
&\quad +(\overline{a} \cdot ((b \cdot (f_{\overline{a},b} + g_{\overline{a},b})) + (\overline{b} \cdot (f_{\overline{a},\overline{b}} + g_{\overline{a},\overline{b}})))) \\
&= (a \cdot ((b \cdot (1+1)) + (\overline{b} \cdot (0+1)))) \\
&\quad +(\overline{a} \cdot ((b \cdot (0+1)) + (\overline{b} \cdot (0+0)))) \\
&= (a \cdot ((b \cdot 1) + (\overline{b} \cdot 1))) + (\overline{a} \cdot ((b \cdot 1) + (\overline{b} \cdot 0)))
\end{aligned}
$$

Binary Operation by Way of Shannon Expansion

$$
\begin{aligned}
f(a,b) &= a \cdot b \\
g(a,b) &= a + b \\
f + g &= (a \cdot b) + (a + b) = (a \cdot b) + a + b = a + b \\
f + g &= (a \cdot (f_a + g_a)) + (\overline{a} \cdot (f_{\overline{a}} + g_{\overline{a}})) \\
&= (a \cdot ((b \cdot (f_{a,b} + g_{a,b})) + (\overline{b} \cdot (f_{a,\overline{b}} + g_{a,\overline{b}})))) \\
&\quad +(\overline{a} \cdot ((b \cdot (f_{\overline{a},b} + g_{\overline{a},b})) + (\overline{b} \cdot (f_{\overline{a},\overline{b}} + g_{\overline{a},\overline{b}})))) \\
&= (a \cdot ((b \cdot (1 + 1)) + (\overline{b} \cdot (0 + 1)))) \\
&\quad +(\overline{a} \cdot ((b \cdot (0 + 1)) + (\overline{b} \cdot (0 + 0)))) \\
&= (a \cdot ((b \cdot 1) + (\overline{b} \cdot 1))) + (\overline{a} \cdot ((b \cdot 1) + (\overline{b} \cdot 0))) \\
&= (a \cdot (b + \overline{b})) + (\overline{a} \cdot (b + 0))
\end{aligned}
$$

Binary Operation by Way of Shannon Expansion

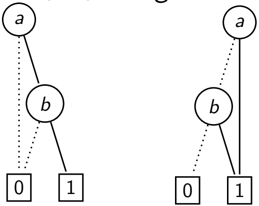
$$
\begin{aligned}
f(a,b) &= a \cdot b \\
g(a,b) &= a + b \\
f + g &= (a \cdot b) + (a + b) = (a \cdot b) + a + b = a + b \\
f + g &= (a \cdot (f_a + g_a)) + (\overline{a} \cdot (f_{\overline{a}} + g_{\overline{a}})) \\
&= (a \cdot ((b \cdot (f_{a,b} + g_{a,b})) + (\overline{b} \cdot (f_{a,\overline{b}} + g_{a,\overline{b}})))) \\
&\quad +(\overline{a} \cdot ((b \cdot (f_{\overline{a},b} + g_{\overline{a},b})) + (\overline{b} \cdot (f_{\overline{a},\overline{b}} + g_{\overline{a},\overline{b}})))) \\
&= (a \cdot ((b \cdot (1 + 1)) + (\overline{b} \cdot (0 + 1)))) \\
&\quad +(\overline{a} \cdot ((b \cdot (0 + 1)) + (\overline{b} \cdot (0 + 0)))) \\
&= (a \cdot ((b \cdot 1) + (\overline{b} \cdot 1))) + (\overline{a} \cdot ((b \cdot 1) + (\overline{b} \cdot 0))) \\
&= (a \cdot (b + \overline{b})) + (\overline{a} \cdot (b + 0)) \\
&= (a \cdot 1) + (\overline{a} \cdot b)
\end{aligned}
$$

Binary Operation by Way of Shannon Expansion

$$
\begin{aligned}
f(a,b) &= a \cdot b \\
g(a,b) &= a + b \\
f + g &= (a \cdot b) + (a + b) = (a \cdot b) + a + b = a + b \\
f + g &= (a \cdot (f_a + g_a)) + (\overline{a} \cdot (f_{\overline{a}} + g_{\overline{a}})) \\
&= (a \cdot ((b \cdot (f_{a,b} + g_{a,b})) + (\overline{b} \cdot (f_{a,\overline{b}} + g_{a,\overline{b}})))) \\
&\quad + (\overline{a} \cdot ((b \cdot (f_{\overline{a},b} + g_{\overline{a},b})) + (\overline{b} \cdot (f_{\overline{a},\overline{b}} + g_{\overline{a},\overline{b}})))) \\
&= (a \cdot ((b \cdot (1 + 1)) + (\overline{b} \cdot (0 + 1)))) \\
&\quad + (\overline{a} \cdot ((b \cdot (0 + 1)) + (\overline{b} \cdot (0 + 0)))) \\
&= (a \cdot ((b \cdot 1) + (\overline{b} \cdot 1))) + (\overline{a} \cdot ((b \cdot 1) + (\overline{b} \cdot 0))) \\
&= (a \cdot (b + \overline{b})) + (\overline{a} \cdot (b + 0)) \\
&= (a \cdot 1) + (\overline{a} \cdot b) \\
&= a + (\overline{a} \cdot b) = a + b
\end{aligned}
$$

Binary Operation by Way of Shannon Expansion

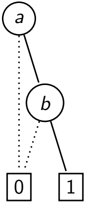
$f = a \cdot b$ $\qquad$ $g = a + b$



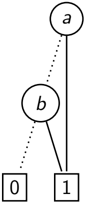
Binary Operation by Way of Shannon Expansion

$f + g$

$f = a \cdot b$

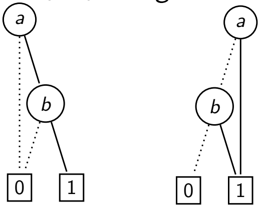


$g = a + b$

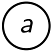


Binary Operation by Way of Shannon Expansion

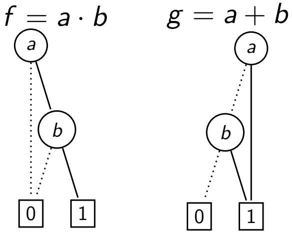
$f = a \cdot b$ $\quad$ $g = a + b$

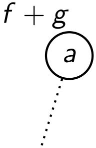


$f + g$



Binary Operation by Way of Shannon Expansion

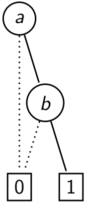




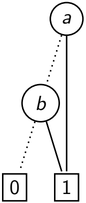
Shannon expansion by $\overline{a}$: $0 + b$

Binary Operation by Way of Shannon Expansion

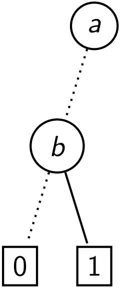
$f = a \cdot b$



$g = a + b$

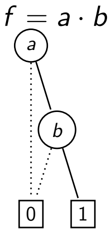


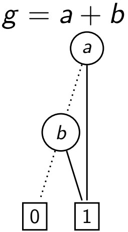
$f + g$

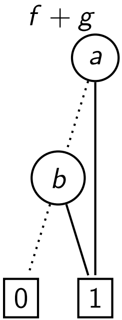


Shannon expansion by $\overline{a}$: $0 + b$

Binary Operation by Way of Shannon Expansion



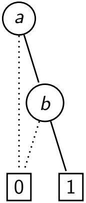




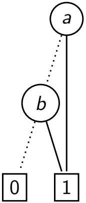
Shannon expansion by a: $b + 1$

Binary Operation by Way of Shannon Expansion

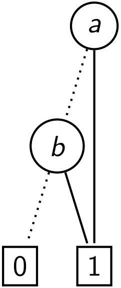
$f = a \cdot b$



$g = a + b$



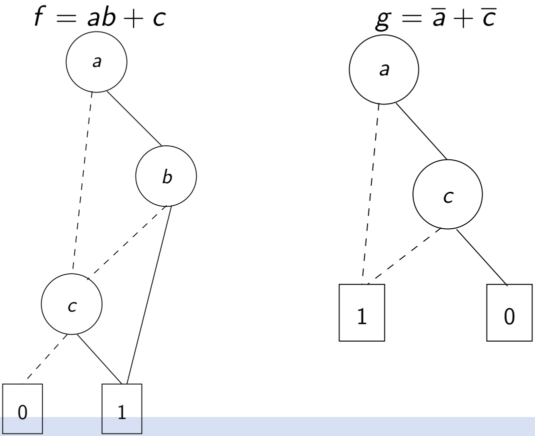
$f + g$



Binary operations are of quadratic complexity in the number of variables.

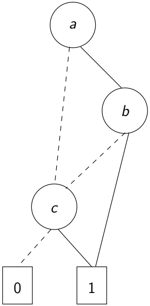
Binary Operation by Way of Shannon Expansion

$$
\begin{aligned}
f &= ab + c \\
g &= \overline{a} + \overline{c} \\
f \cdot g &= (ab + c) \cdot (\overline{a} + \overline{c}) = \overline{a}c + ab\overline{c}
\end{aligned}
$$

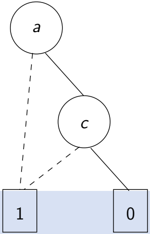


Binary Operation by Way of Shannon Expansion

$$f = ab + c$$



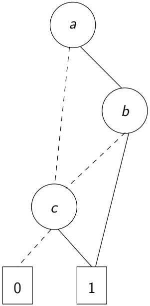
$$g = \overline{a} + \overline{c}$$



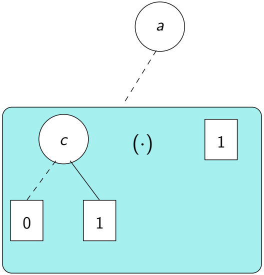
$$f \cdot g = \overline{a}c + ab\overline{c}$$

Binary Operation by Way of Shannon Expansion

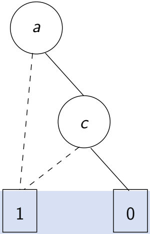
$$f = ab + c$$



$$f \cdot g = \overline{a}c + ab\overline{c}$$

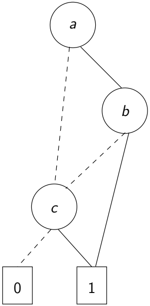


$$g = \overline{a} + \overline{c}$$

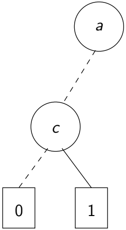


Binary Operation by Way of Shannon Expansion

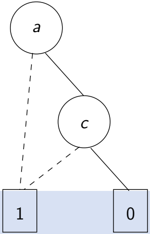
$f = ab + c$



$f \cdot g = \overline{a}c + ab\overline{c}$

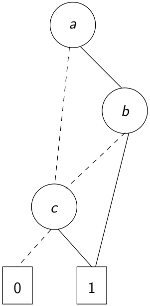


$g = \overline{a} + \overline{c}$

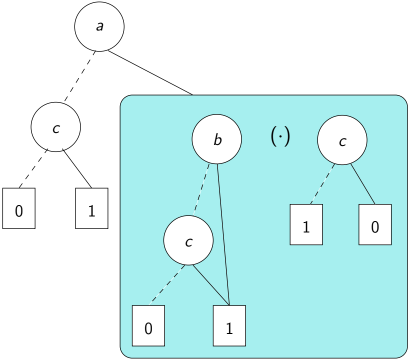


Binary Operation by Way of Shannon Expansion

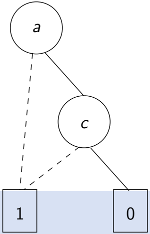
$f = ab + c$



$f \cdot g = \overline{a}c + ab\overline{c}$

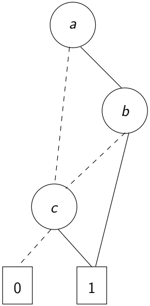


$g = \overline{a} + \overline{c}$

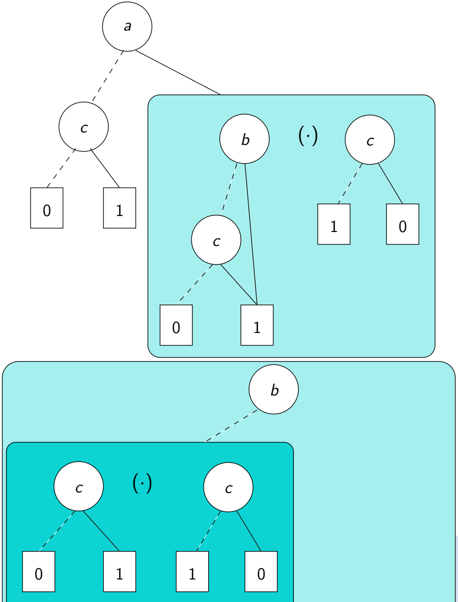


Binary Operation by Way of Shannon Expansion

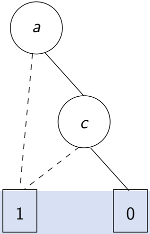
$f = ab + c$



$f \cdot g = \overline{a}c + ab\overline{c}$

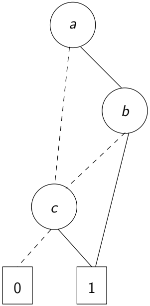


$g = \overline{a} + \overline{c}$

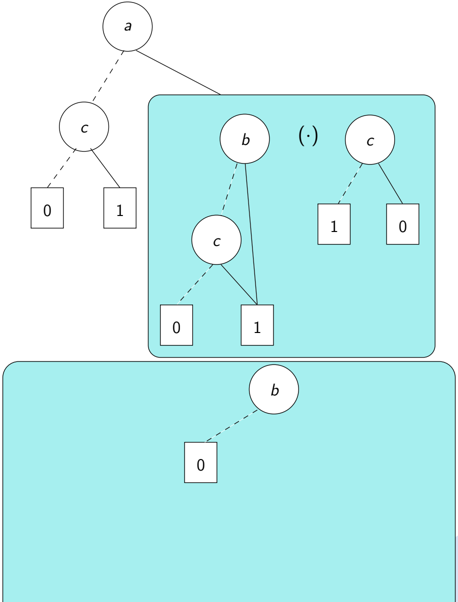


Binary Operation by Way of Shannon Expansion

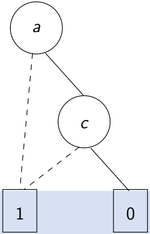
$f = ab + c$



$f \cdot g = \overline{a}c + ab\overline{c}$

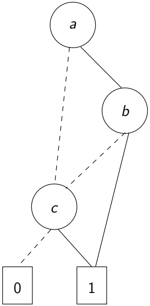


$g = \overline{a} + \overline{c}$

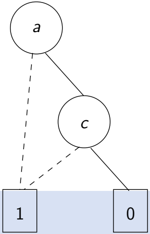


Binary Operation by Way of Shannon Expansion

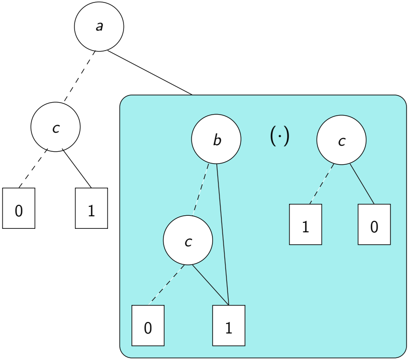
$$f = ab + c$$

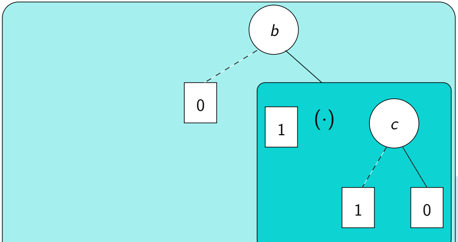


$$g = \overline{a} + \overline{c}$$



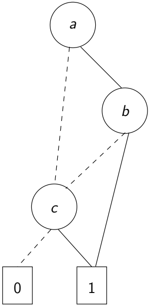
$$f \cdot g = \overline{a}c + ab\overline{c}$$



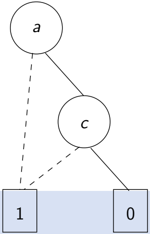


Binary Operation by Way of Shannon Expansion

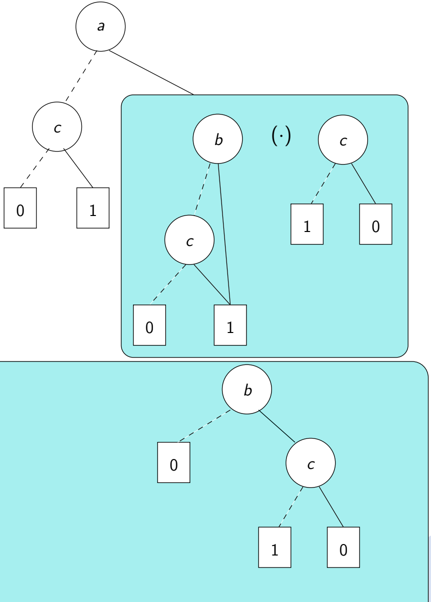
$f = ab + c$



$g = \overline{a} + \overline{c}$

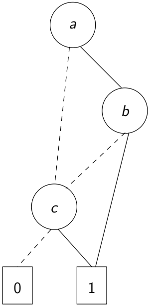


$f \cdot g = \overline{a}c + ab\overline{c}$

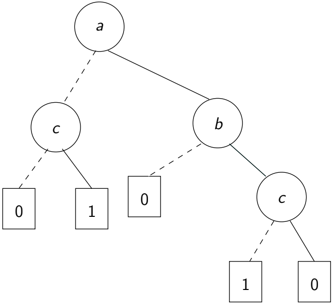


Binary Operation by Way of Shannon Expansion

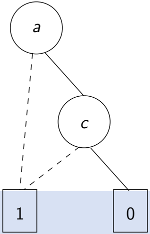
$$f = ab + c$$



$$f \cdot g = \overline{a}c + ab\overline{c}$$

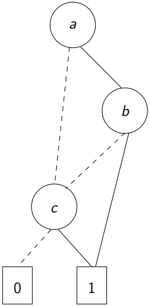


$$g = \overline{a} + \overline{c}$$

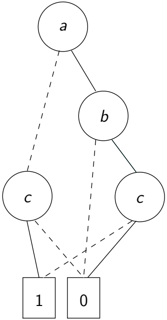


Binary Operation by Way of Shannon Expansion

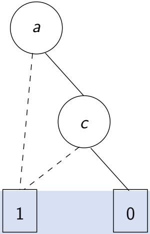
$f = ab + c$



$f \cdot g = \overline{a}c + ab\overline{c}$



$g = \overline{a} + \overline{c}$



ROBDD - Complexity of Operations

Operation	SOP	POS	ROBDD
Conjunction	Quadratic: n^2	Constant	Quadratic: m^2
Disjunction	Constant	Quadratic: n^2	Quadratic: m^2
Complement	**Exponential**: m^n	**Exponential**: m^n	Constant
SAT	Constant	**Exponential**: 2^m	Constant
TAUTOLOGY	**Exponential**: 2^m	Constant	Constant

Logic Synthesis

SoP $\leftrightarrow$ PoS Conversion

$$f_{\mathsf{SoP}} \xRightarrow{\text{Double Complement}} \xRightarrow{\text{DeMorgan}} \xRightarrow{\text{Distributive Law}} \xRightarrow{\text{DeMorgan}} f_{\mathsf{PoS}}$$

SoP $\leftrightarrow$ PoS Conversion

$$f_{\mathrm{SoP}} \overset{\text{Double Complement}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} \overset{\text{Distributive Law}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} f_{\mathrm{PoS}}$$

$$f(a,b,c,d) = (a \cdot b) + (c \cdot d)$$

SoP $\leftrightarrow$ PoS Conversion

$$f_{\text{SoP}} \overset{\text{Double Complement}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} \overset{\text{Distributive Law}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} f_{\text{PoS}}$$

$$\begin{aligned} f(a,b,c,d) &= (a \cdot b) + (c \cdot d) \\ &= \overline{\overline{a \cdot b + c \cdot d}} \end{aligned}$$

SoP $\leftrightarrow$ PoS Conversion

$$f_{\text{SoP}} \overset{\text{Double Complement}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} \overset{\text{Distributive Law}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} f_{\text{PoS}}$$

$$\begin{aligned}
f(a,b,c,d) &= (a\cdot b)+(c\cdot d)\\
&= \overline{\overline{a\cdot b+c\cdot d}}\\
&= \overline{\overline{a\cdot b}\cdot\overline{c\cdot d}}
\end{aligned}$$

SoP $\leftrightarrow$ PoS Conversion

$$f_{\text{SoP}} \xRightarrow{\text{Double Complement}} \xRightarrow{\text{DeMorgan}} \xRightarrow{\text{Distributive Law}} \xRightarrow{\text{DeMorgan}} f_{\text{PoS}}$$

$$\begin{aligned}
f(a,b,c,d) &= (a \cdot b) + (c \cdot d) \\
&= \overline{\overline{a \cdot b + c \cdot d}} \\
&= \overline{\overline{a \cdot b} \cdot \overline{c \cdot d}} \\
&= \overline{(\overline{a} + \overline{b}) \cdot (\overline{c} + \overline{d})}
\end{aligned}$$

SoP $\leftrightarrow$ PoS Conversion

$$f_{\text{SoP}} \overset{\text{Double Complement}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} \overset{\text{Distributive Law}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} f_{\text{PoS}}$$

$$\begin{aligned}
f(a,b,c,d) &= (a \cdot b) + (c \cdot d) \\
&= \overline{\overline{a \cdot b + c \cdot d}} \\
&= \overline{\overline{a \cdot b} \cdot \overline{c \cdot d}} \\
&= \overline{(\overline{a} + \overline{b}) \cdot (\overline{c} + \overline{d})} \\
&= \overline{(\overline{a} \cdot \overline{c}) + (\overline{a} \cdot \overline{d}) + (\overline{b} \cdot \overline{c}) + (\overline{b} \cdot \overline{d})}
\end{aligned}$$

SoP $\leftrightarrow$ PoS Conversion

$$f_{\text{SoP}} \xRightarrow{\text{Double Complement}} \xRightarrow{\text{DeMorgan}} \xRightarrow{\text{Distributive Law}} \xRightarrow{\text{DeMorgan}} f_{\text{PoS}}$$

$$\begin{aligned}
f(a,b,c,d) &= (a\cdot b)+(c\cdot d)\\
&= \overline{\overline{a\cdot b + c\cdot d}}\\
&= \overline{\overline{a\cdot b}\cdot\overline{c\cdot d}}\\
&= \overline{(\overline{a}+\overline{b})\cdot(\overline{c}+\overline{d})}\\
&= \overline{(\overline{a}\cdot\overline{c})+(\overline{a}\cdot\overline{d})+(\overline{b}\cdot\overline{c})+(\overline{b}\cdot\overline{d})}\\
&= \overline{(\overline{a}\cdot\overline{c})}\cdot\overline{(\overline{a}\cdot\overline{d})}\cdot\overline{(\overline{b}\cdot\overline{c})}\cdot\overline{(\overline{b}\cdot\overline{d})}
\end{aligned}$$

SoP $\leftrightarrow$ PoS Conversion

$$f_{\text{SoP}} \overset{\text{Double Complement}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} \overset{\text{Distributive Law}}{\Longrightarrow} \overset{\text{DeMorgan}}{\Longrightarrow} f_{\text{PoS}}$$

$$\begin{aligned}
f(a,b,c,d) &= (a \cdot b) + (c \cdot d) \\
&= \overline{\overline{a \cdot b + c \cdot d}} \\
&= \overline{\overline{a \cdot b} \cdot \overline{c \cdot d}} \\
&= \overline{(\overline{a} + \overline{b}) \cdot (\overline{c} + \overline{d})} \\
&= \overline{(\overline{a} \cdot \overline{c}) + (\overline{a} \cdot \overline{d}) + (\overline{b} \cdot \overline{c}) + (\overline{b} \cdot \overline{d})} \\
&= \overline{(\overline{a} \cdot \overline{c})} \cdot \overline{(\overline{a} \cdot \overline{d})} \cdot \overline{(\overline{b} \cdot \overline{c})} \cdot \overline{(\overline{b} \cdot \overline{d})} \\
&= (a + c) \cdot (a + d) \cdot (b + c) \cdot (b + d)
\end{aligned}$$

SoP $\leftrightarrow$ PoS Conversion

$$f_{\mathsf{SoP}} \xRightarrow{\text{Double Complement}} \xRightarrow{\text{DeMorgan}} \xRightarrow{\text{Distributive Law}} \xRightarrow{\text{DeMorgan}} f_{\mathsf{PoS}}$$

$$\begin{aligned}
f(a,b,c,d) &= (a\cdot b)+(c\cdot d) \\
&= \overline{\overline{a\cdot b + c\cdot d}} \\
&= \overline{\overline{a\cdot b}\cdot\overline{c\cdot d}} \\
&= \overline{(\overline{a}+\overline{b})\cdot(\overline{c}+\overline{d})} \\
&= \overline{(\overline{a}\cdot\overline{c})+(\overline{a}\cdot\overline{d})+(\overline{b}\cdot\overline{c})+(\overline{b}\cdot\overline{d})} \\
&= \overline{(\overline{a}\cdot\overline{c})}\cdot\overline{(\overline{a}\cdot\overline{d})}\cdot\overline{(\overline{b}\cdot\overline{c})}\cdot\overline{(\overline{b}\cdot\overline{d})} \\
&= (a+c)\cdot(a+d)\cdot(b+c)\cdot(b+d)
\end{aligned}$$

SoP $\leftrightarrow$ PoS Conversion is of exponential complexity: m^n.

Boolean formula $\leftrightarrow$ BDD Conversion

A Boolean function is converted to a ROBDD by starting with a variable representation and recursively applying the Boolean operators.

Some Boolean functions have ROBDD of exponential size.

Boolean formula $\leftrightarrow$ BDD Conversion is of exponential complexity.

Logic Synthesis

Two-level Minimization

- Karnaugh-Veitch Diagrams: Limited to 4 variables;
- Quine-McCluskey

Logic Synthesis

Multi-level Minimization

- Two-Level Minimization is limited to relatively small functions.
- Fan-in limitations of gates, depending on technology

Multi-level Minimization

- Two-Level Minimization is limited to relatively small functions.
- Fan-in limitations of gates, depending on technology

Multi-level minimization uses

- Factoring
- Decomposition
- Extraction
- Substitution
- Elimination

Factoring

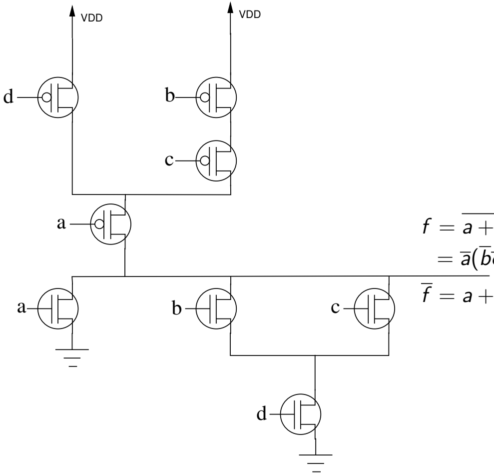
- A factored form of a Boolean formula is a tree representation with the nodes being AND-operations, OR-operations or literals.
- Factoring reformulates a Boolean function in a form with minimum number of literals, because an implementation with complex CMOS gates in general requires $2n$ transistors for a factored form with n literals

$$F = ac + ad + bc + bd + e$$

can be factored into

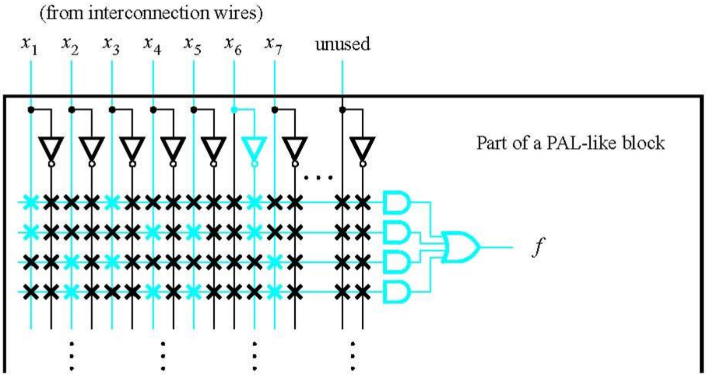
$$F = (a + b)(c + d) + e$$

CMOS Gate of a Factored Form



$$f = \overline{a + (b+c)d}$$
$$= \overline{a}(\overline{b}\,\overline{c} + \overline{d})$$
$$\overline{f} = a + (b+c)d$$

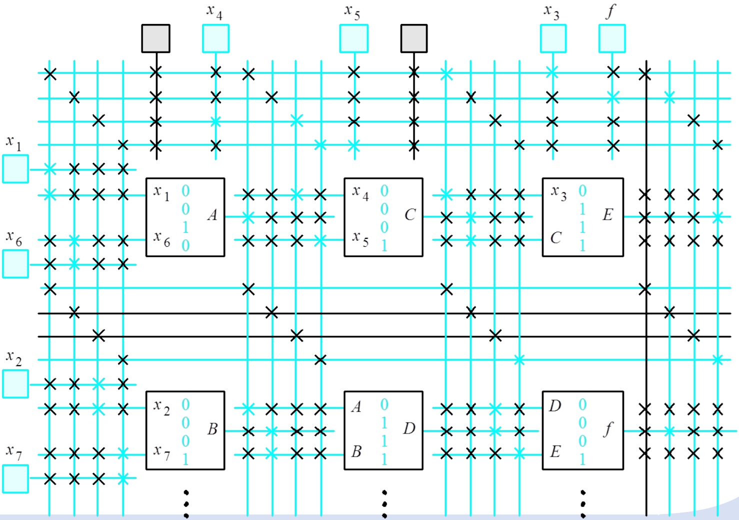
Consider $f = x_1 x_3 \overline{x_6} + x_1 x_4 x_5 \overline{x_6} + x_2 x_3 x_7 + x_2 x_4 x_5 x_7$



Implementation in an FPGA with 2-input LUTs is not suitable.
Transform f as follows:

$$\begin{aligned} f &= x_1 x_3 \overline{x_6} + x_1 x_4 x_5 \overline{x_6} + x_2 x_3 x_7 + x_2 x_4 x_5 x_7 \\ &= x_1 \overline{x_6}(x_3 + x_4 x_5) + x_2 x_7 (x_3 + x_4 x_5) \\ &= (x_3 + x_4 x_5)(x_1 \overline{x_6} + x_2 x_7) \end{aligned}$$

$$f = (x_3 + x_4 x_5)(x_1 \overline{x_6} + x_2 x_7)$$

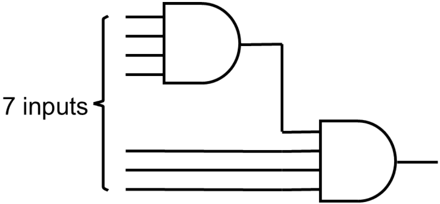


Fan-in Constraints

In all implementation technologies:

- CPLDs
- FPGAs
- ASICs

Large gates can be implemented with smaller gates:



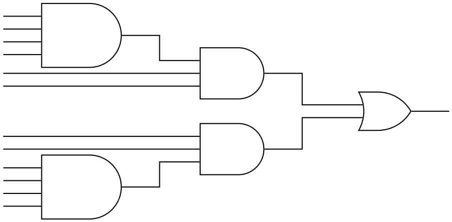
Consider

$$f = x_1\overline{x_2}x_3\overline{x_4}x_5x_6 + x_1x_2\overline{x_3}\,\overline{x_4}\,\overline{x_5}x_6$$

Consider

$$f = x_1\overline{x_2}x_3\overline{x_4}x_5x_6 + x_1x_2\overline{x_3}\,\overline{x_4}\,\overline{x_5}x_6$$

Can be implemented as



with cost=21.

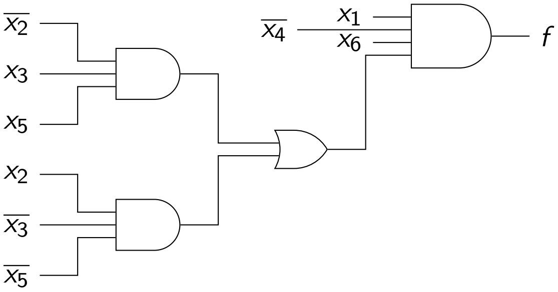
Or by factoring as follows:

$$\begin{aligned} f &= x_1\overline{x_2}x_3\overline{x_4}x_5x_6 + x_1x_2\overline{x_3}\,\overline{x_4}\,\overline{x_5}x_6 \\ &= x_1\overline{x_4}x_6\left(\overline{x_2}x_3x_5 + x_2\overline{x_3}\,\overline{x_5}\right) \end{aligned}$$

Or by factoring as follows:

$$
\begin{aligned}
f &= x_1\overline{x_2}x_3\overline{x_4}x_5x_6 + x_1x_2\overline{x_3}\,\overline{x_4}\,\overline{x_5}x_6 \\
&= x_1\overline{x_4}x_6\left(\overline{x_2}x_3x_5 + x_2\overline{x_3}\,\overline{x_5}\right)
\end{aligned}
$$

which can be implemented as



with cost=16.

Decomposition

A function is decomposed into other, simpler functions.

$$F(a,b,c,d) = abc + abd + \overline{a}\overline{c}\overline{d} + \overline{b}\overline{c}\overline{d}$$

$$\begin{aligned} F &= X \cdot Y + \overline{X} \cdot \overline{Y} \\ X &= ab \\ Y &= c + d \end{aligned}$$

Decomposition Example

$$f = \overline{x_1}x_2x_3 + x_1\overline{x_2}x_3 + x_1x_2x_4 + \overline{x_1}\,\overline{x_2}x_4$$

Decomposition Example

$$
\begin{aligned}
f &= \overline{x_1}x_2x_3 + x_1\overline{x_2}x_3 + x_1x_2x_4 + \overline{x_1}\,\overline{x_2}x_4 \\
&= (\overline{x_1}x_2 + x_1\overline{x_2})x_3 + (x_1x_2 + \overline{x_1}\,\overline{x_2})x_4
\end{aligned}
$$

Decomposition Example

$$
\begin{aligned}
f &= \overline{x_1}x_2x_3 + x_1\overline{x_2}x_3 + x_1x_2x_4 + \overline{x_1}\,\overline{x_2}x_4 \\
&= (\overline{x_1}x_2 + x_1\overline{x_2})x_3 + (x_1x_2 + \overline{x_1}\,\overline{x_2})x_4
\end{aligned}
$$

$$
g = \overline{x_1}x_2 + x_1\overline{x_2}
$$

Decomposition Example

$$
\begin{aligned}
f &= \overline{x_1}x_2x_3 + x_1\overline{x_2}x_3 + x_1x_2x_4 + \overline{x_1}\,\overline{x_2}x_4 \\
&= (\overline{x_1}x_2 + x_1\overline{x_2})x_3 + (x_1x_2 + \overline{x_1}\,\overline{x_2})x_4
\end{aligned}
$$

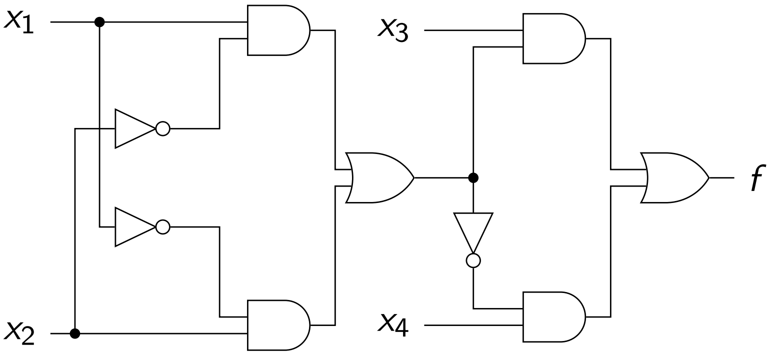
$$
\begin{aligned}
g &= \overline{x_1}x_2 + x_1\overline{x_2} \\
\overline{g} &= x_1x_2 + \overline{x_1}\,\overline{x_2}
\end{aligned}
$$

Decomposition Example

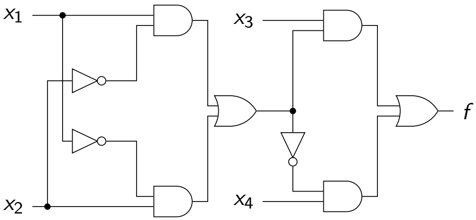
$$
\begin{aligned}
f &= \overline{x_1}x_2x_3 + x_1\overline{x_2}x_3 + x_1x_2x_4 + \overline{x_1}\,\overline{x_2}x_4 \\
&= (\overline{x_1}x_2 + x_1\overline{x_2})x_3 + (x_1x_2 + \overline{x_1}\,\overline{x_2})x_4
\end{aligned}
$$

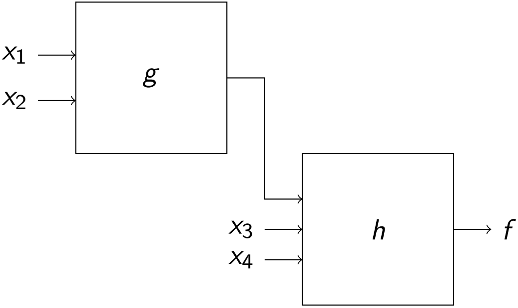
$$
\begin{aligned}
g &= \overline{x_1}x_2 + x_1\overline{x_2} \\
\overline{g} &= x_1x_2 + \overline{x_1}\,\overline{x_2}
\end{aligned}
$$

$$
f = gx_3 + \overline{g}x_4
$$



Logic circuit for $f = gx_3 + \overline{g}x_4$.





$$f = gx_3 + \overline{g}x_4 = h(g(x_1, x_2), x_3, x_4).$$

Logic Transformations

- Logic transformations change the structure of the Boolean network (restructuring).
- Heuristics and cost functions guide the restructuring process.

Logic Synthesis

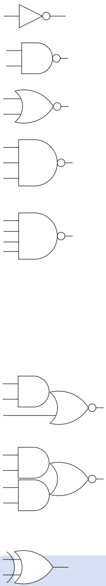
Technology Mapping

- Given a netlist of abstract gates, and a library of technology specific gates, technology mapping creates a technology specific netlist of gates.
- Technology mapping is different for standard cells, gate arrays and FPGAs.
- Only after this step, technology specific information such as gate delay, area, and power consumption is available.

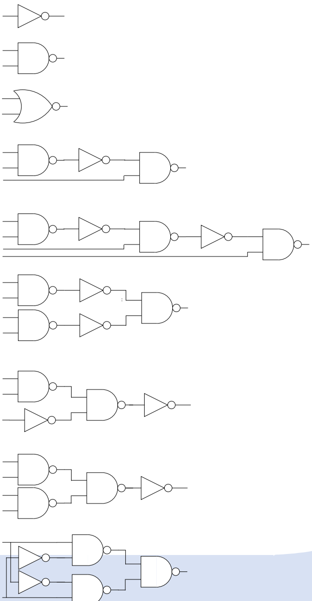
Technology Libraries

Gate	Cost
INV	2
NAND2	3
NOR2	4
NAND3	4
NAND4	5
AIO21	4
AOI22	5
XOR	4

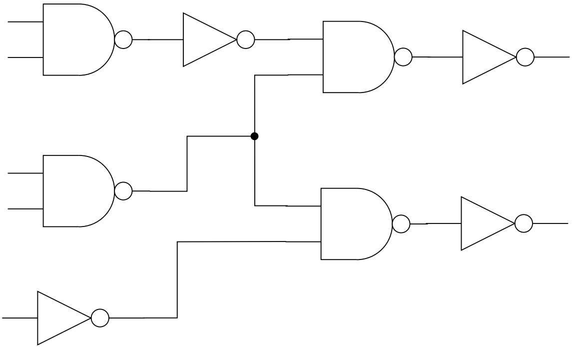
Symbol



Pattern Graph



Subject Graph



The **subject graph** is the standardized netlist representation (e.g. consisting only of NAND2 and INV) of the target design.

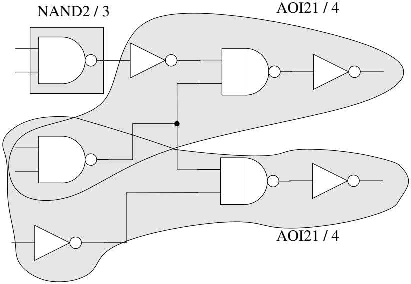
Graph Covering

Graph covering finds a cover of library elements (patterns DAG (Directed Acyclic Graph)) of the subject graph such that

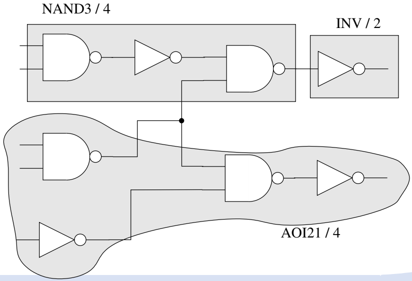
- All gates of the subject graph are covered;
- No internal node of a Pattern DAG is the input of another pattern graph;
- The cost function (area, power, delay, ...) is minimized;
- All design constraints are met.

Graph Covering

Legal covering:



Illegal covering:

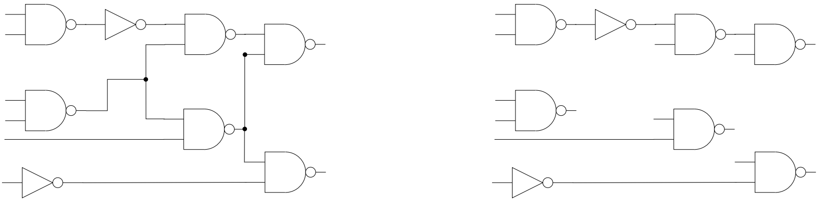


Atomic Pattern Set

Empirically the choice of NAND2 and INV as the only atomic patterns is a good solution.

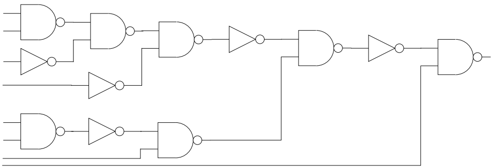
Additional patterns (e.g. AND2, NOR2, NAND3, ...) lead rarely to better solutions, but make the graph covering problem harder.

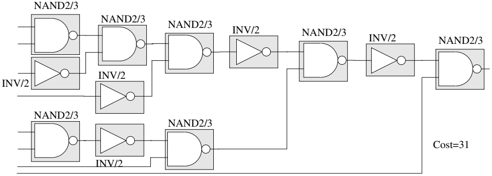
Tree Covering

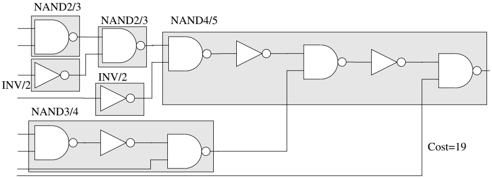


1. Convert the Boolean network into a NAND2-INV netlist to create the subject graph;
2. Partition the subject graph into a forest of trees;
3. Each tree is optimally covered separately:
   1. Generate a complete set of matches for the tree;
   2. Select the best match with a dynamic programming algorithm.

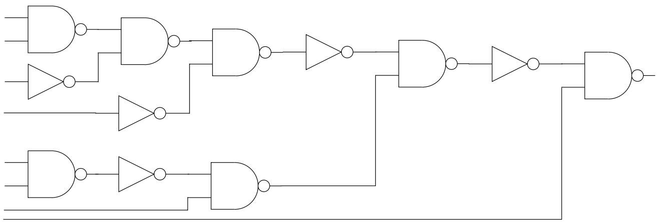
Tree Covering

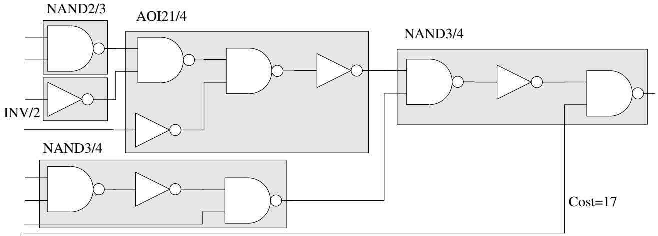




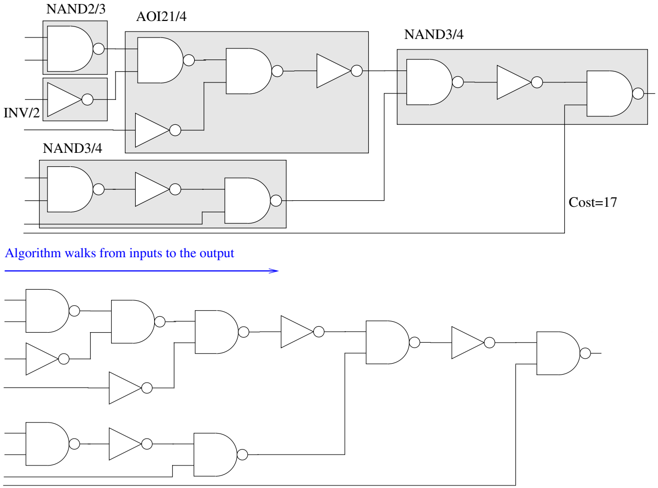


Tree Covering



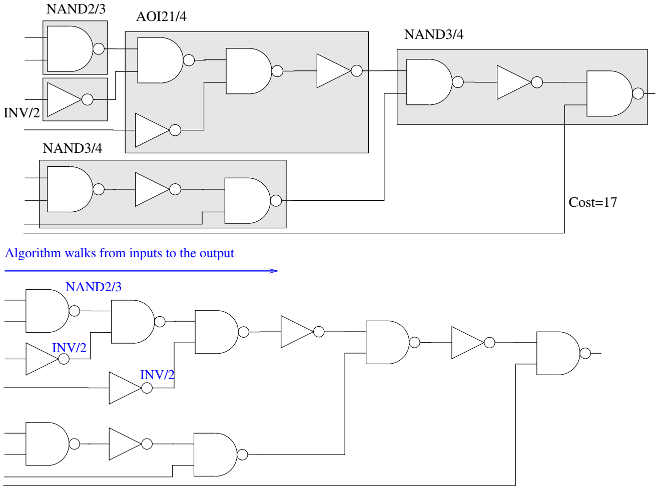


Tree Covering



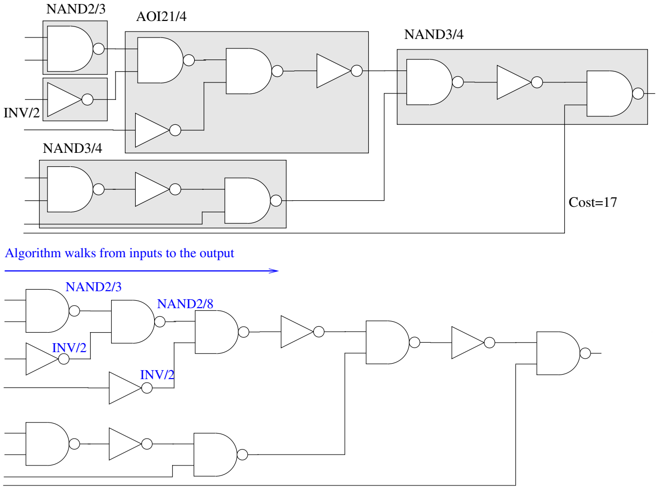
Optimal tree covering based on recursive, optimal sub-tree covering.

Tree Covering



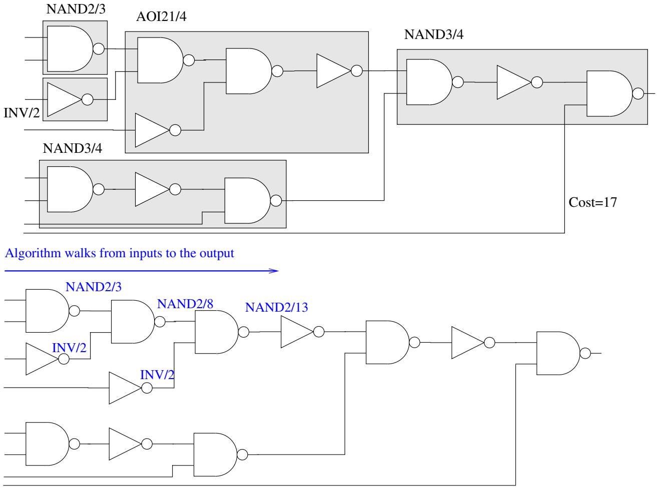
Optimal tree covering based on recursive, optimal sub-tree covering.

Tree Covering



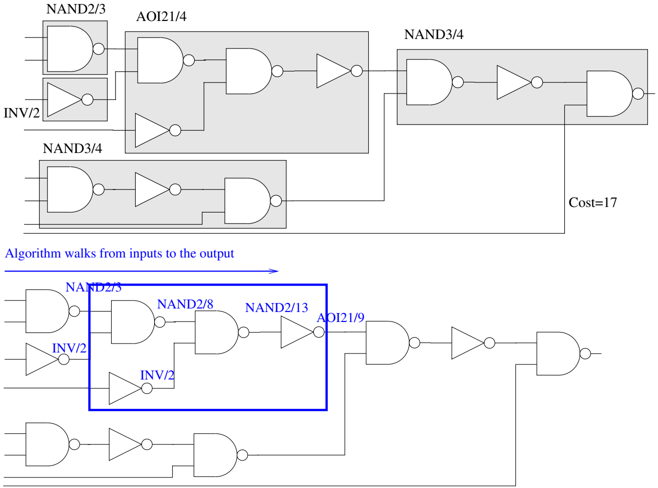
Optimal tree covering based on recursive, optimal sub-tree covering.

Tree Covering



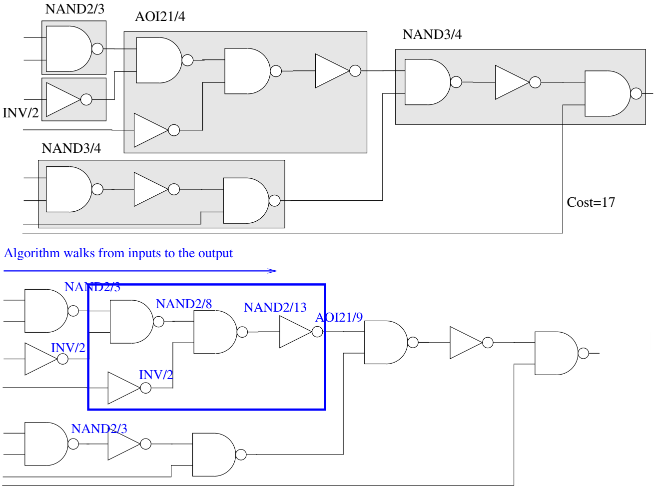
Optimal tree covering based on recursive, optimal sub-tree covering.

Tree Covering



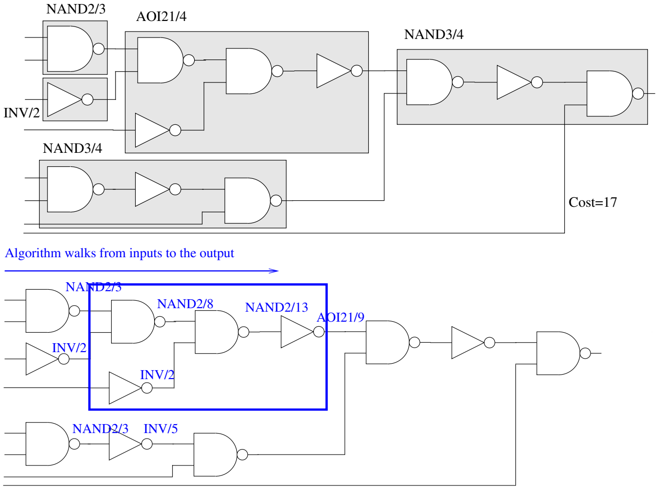
Optimal tree covering based on recursive, optimal sub-tree covering.

Tree Covering



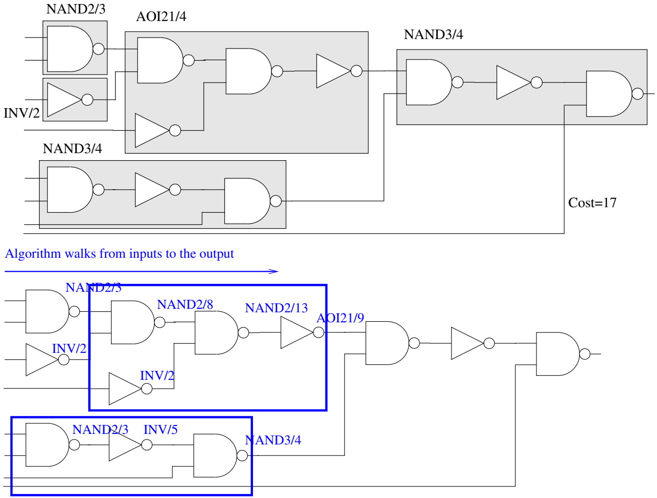
Optimal tree covering based on recursive, optimal sub-tree covering.

Tree Covering



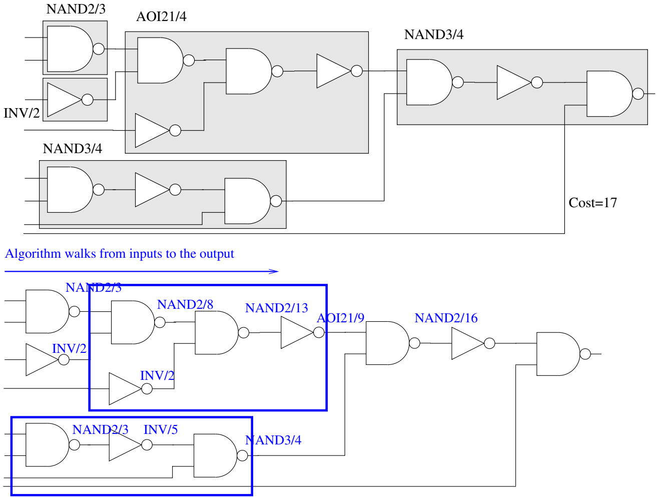
Optimal tree covering based on recursive, optimal sub-tree covering.

Tree Covering



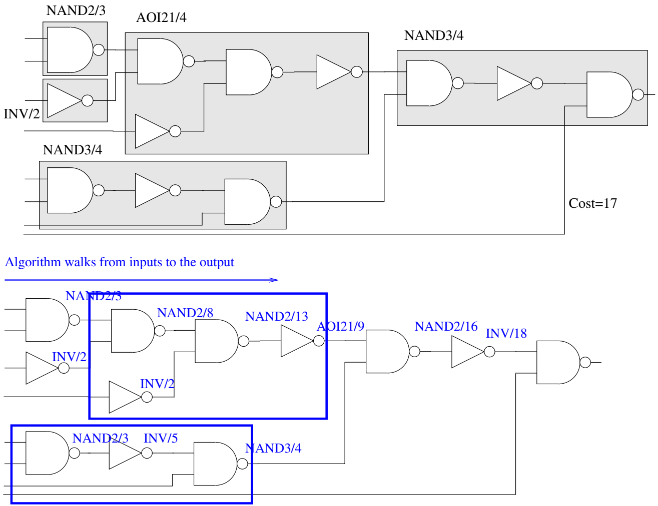
Optimal tree covering based on recursive, optimal sub-tree covering.

Tree Covering



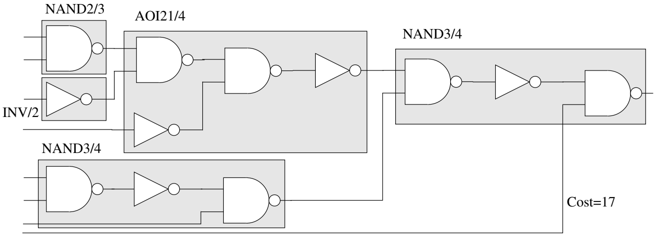
Optimal tree covering based on recursive, optimal sub-tree covering.

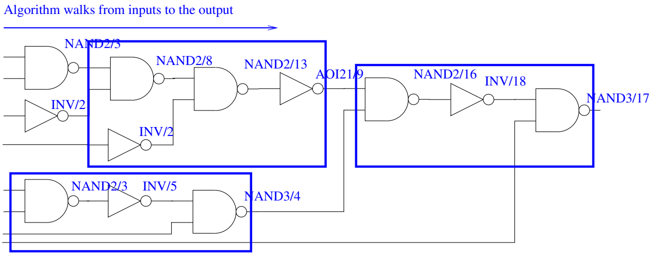
Tree Covering



Optimal tree covering based on recursive, optimal sub-tree covering.

Tree Covering





Optimal tree covering based on recursive, optimal sub-tree covering.

Summary

- Representation of Boolean functions
- Two-level optimization
- Multi-level optimization
- Technology mapping