Fast Extract with Cube Hashing (FXCH)

B. Schmitt[1] A. Mishchenko[2] V. Kravets[3] R. Brayton[2]
A. Reis[1]

[1] Institute of Informatics
Federal University of Rio Grande do Sul (UFRGS)

[2] Department of EECS
University of California, Berkeley

[3] IBM Thomas J. Watson Research Center
Yorktown Heights, NY

ASP-DAC, January 2017

Table of Contents

Table of Contents

The Bird's Eye View

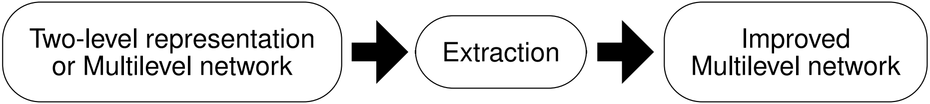
Uses

- Reduce the complexity of an available multilevel network.
- Construct a multilevel network from a two-level representation of a Boolean function.

The Bird's Eye View

Uses

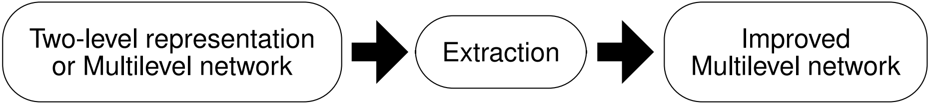
- Reduce the complexity of an available multilevel network.
- Construct a multilevel network from a two-level representation of a Boolean function.



The Bird's Eye View

Uses

- Reduce the complexity of an available multilevel network.
- Construct a multilevel network from a two-level representation of a Boolean function.



Motivation

- Original fast-extract (*fx*) is old.
- Since its appearance:
  - the transistor count has increased by three orders of magnitude
  - the price of memory decreased by four orders of magnitude

Table of Contents

Boolean functions - Two-level Representation

- Any Boolean function can be represented as a truth table.

Boolean functions - Two-level Representation

- Any Boolean function can be represented as a truth table.

Truth table

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

minterms

Boolean functions - Two-level Representation

- Any Boolean function can be represented as a truth table.

Truth table

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

minterms

On-set

Boolean functions - Two-level Representation

- Any Boolean function can be represented as a truth table.

Truth table

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

minterms

On-set

x_1	x_2	x_3	y_1
0	-	1	1
0	1	-	1
-	1	1	1

Boolean functions - Two-level Representation

- Any Boolean function can be represented as a truth table.

Truth table

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

minterms

On-set

x_1	x_2	x_3	y_1
0	-	1	1
0	1	-	1
-	1	1	1

Off-set

Boolean functions - Two-level Representation

- Any Boolean function can be represented as a truth table.

Truth table

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

minterms

On-set

x_1	x_2	x_3	y_1
0	-	1	1
0	1	-	1
-	1	1	1

Off-set

x_1	x_2	x_3	y_1
-	0	0	0
1	0	-	0
1	-	0	0

Boolean functions - Two-level Representation

- Any Boolean function can be represented as a two-level sum of products (SOP), which is a Boolean OR of implicants (i.e. $S = c_1 + c + \cdots + c_n$)

Truth table

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

minterms

On-set

x_1	x_2	x_3	y_1
0	-	1	1
0	1	-	1
-	1	1	1

Off-set

x_1	x_2	x_3	y_1
-	0	0	0
1	0	-	0
1	-	0	0

Boolean functions - Two-level Representation

- Any Boolean function can be represented as a two-level sum of products (SOP), which is a Boolean OR of implicants (i.e. $S = c_1 + c + \cdots + c_n$)

Truth table

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

minterms

On-set

x_1	x_2	x_3	y_1
0	-	1	1
0	1	-	1
-	1	1	1

$$f = \overline{x}_1 x_3 + \overline{x}_1 x_2 + x_2 x_3$$

Off-set

x_1	x_2	x_3	y_1
-	0	0	0
1	0	-	0
1	-	0	0

Boolean functions - Two-level Representation

- Any Boolean function can be represented as a two-level sum of products (SOP), which is a Boolean OR of implicants (i.e. $S = c_1 + c + \cdots + c_n$)

Truth table

x_1	x_2	x_3	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

minterms

On-set

x_1	x_2	x_3	y_1
0	-	1	1
0	1	-	1
-	1	1	1

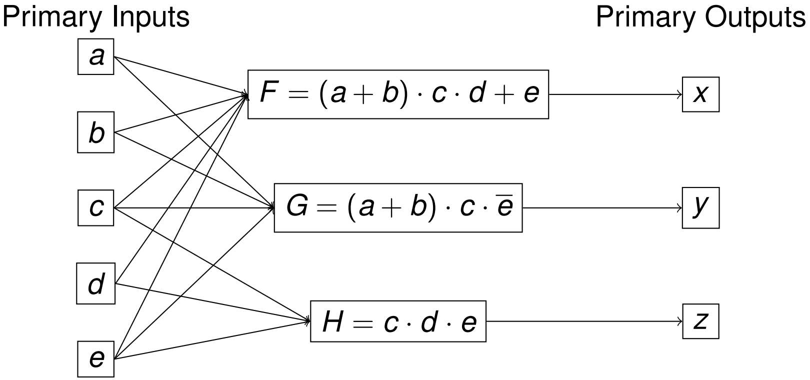
$$f = \overline{x}_1 x_3 + \overline{x}_1 x_2 + x_2 x_3$$

Off-set

x_1	x_2	x_3	y_1
-	0	0	0
1	0	-	0
1	-	0	0

$$\overline{f} = \overline{x}_2 \overline{x}_3 + x_1 \overline{x}_2 + x_1 \overline{x}_3$$

Boolean Networks



Optimization of Boolean Networks

Optimization of Boolean Networks - Overview

The Goal

Obtain an equivalent representation of a Boolean network optimal with respect to some design constraints.

Optimization of Boolean Networks - Overview

The Goal

Obtain an equivalent representation of a Boolean network optimal with respect to some design constraints.

- Typical constraints:
  - Area
  - Delay

Optimization of Boolean Networks - Overview

The Goal

Obtain an equivalent representation of a Boolean network optimal with respect to some design constraints.

- Typical constraints:
  - Area
  - Delay

In multilevel logic minimal-area implementations generally don't correspond to minimal delay ones and vice versa.

Optimization of Boolean Networks - Overview

The Goal

Obtain an equivalent representation of a Boolean network optimal with respect to some design constraints.

- Typical constraints:
  - Area
  - Delay

In multilevel logic minimal-area implementations generally don't correspond to minimal delay ones and vice versa.

- Truly multiple-objective optimization problem.
- Exact methods are, generally, impractical even for a medium-size network.

Optimization of Boolean Networks - Heuristic Methods

Use of heuristic methods which improve the network through **logic transformations** that preserve the input/output network behavior

Optimization of Boolean Networks - Heuristic Methods

Use of heuristic methods which improve the network through **logic transformations** that preserve the input/output network behavior

- All modern synthesis systems are transformation based.
- Logic transformations
  - Most are defined so that network equivalence is guaranteed and does not need to be checked.
  - Virtually impossible to claim that all equivalent networks can be explored by applying some sequence of transformations.
  - Local optimums
- Five key transformations: decompositions, **extraction**, factoring, substitution, and elimination.

Extraction

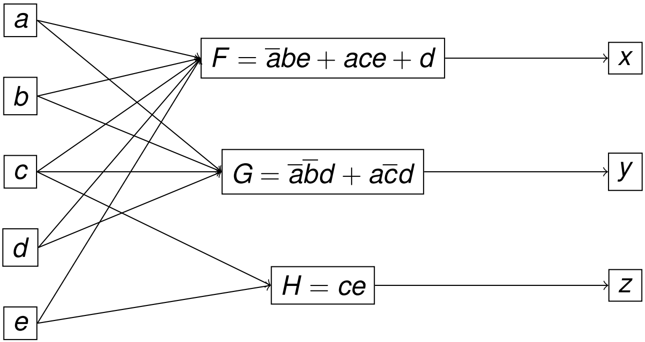
Definition

Extraction is the process of identifying common sub-expressions and using them to create new intermediate functions, which are associated with new variables, and re-expressing the original functions in term of the original as well as the new variables

Extraction

Definition

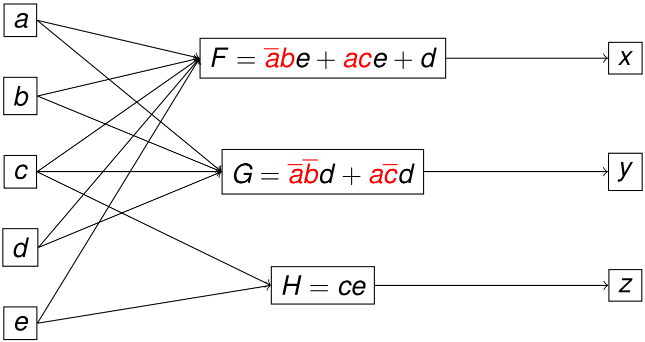
Extraction is the process of identifying common sub-expressions and using them to create new intermediate functions, which are associated with new variables, and re-expressing the original functions in term of the original as well as the new variables



Extraction

Definition

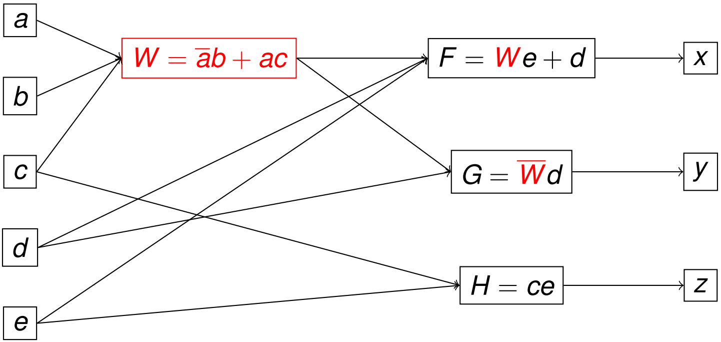
Extraction is the process of identifying common sub-expressions and using them to create new intermediate functions, which are associated with new variables, and re-expressing the original functions in term of the original as well as the new variables



Extraction

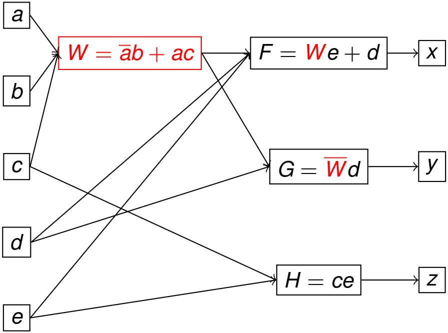
Definition

Extraction is the process of identifying common sub-expressions and using them to create new intermediate functions, which are associated with new variables, and re-expressing the original functions in term of the original as well as the new variables



Algebraic Division

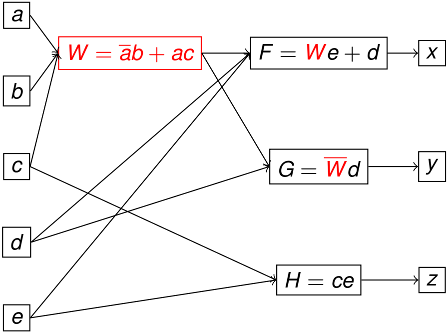
- **Algebraic**: Treats Boolean functions as polynomials. enables the optimization of logic networks through the use of general properties of polynomial algebra.



- $\overline{a}b + ac$ is said to be a kernel of F and G.
- In F, e is the co-kernel of $\overline{a}b + ac$.
- In G, d is the co-kernel of $\overline{a}b + ac$.

Algebraic Division

- **Algebraic**: Treats Boolean functions as polynomials. enables the optimization of logic networks through the use of general properties of polynomial algebra.



- $\bar{a}b + ac$ is said to be a kernel of F and G.
- In F, e is the co-kernel of $\bar{a}b + ac$.
- In G, d is the co-kernel of $\bar{a}b + ac$.

Fundamental Theorem

Given two expressions f and g, and their respective set of kernels $K(f)$ and $K(g)$, f and g have a multiple-cube common divisor if and only if there exists kernels $k_f \in K(f)$ and $k_g \in K(g)$ such that $k_f \cap k_g$ has two or more terms, i.e $k_f \cap k_g$ is not a single cube.

Extraction

1. Enumerate all **kernels** from all functions.
2. Choose a “good” kernel intersection.
3. Create a new node with this as a function.
4. Substitute this new node in the functions that have it as function.
5. Repeat 1, 2, 3 and 4 until no more “good” kernel intersection is found.

Extraction

1. Enumerate all **kernels** from all functions.
2. Choose a “good” kernel intersection.
3. Create a new node with this as a function.
4. Substitute this new node in the functions that have it as function.
5. Repeat 1, 2, 3 and 4 until no more “good” kernel intersection is found.

- Re-computation of kernels after every substitution (expensive).
- Some function have a very large set of kernels.
- Cannot identify if a kernel can be as complemented node.

Fast-extract

Compute a subset of kernels.

- Double-cube kernel extraction [Rajski et al ‘90] (***fast-extract***)
  - single-cube double-literal kernels.
  - double-cube divisors.

Fast-extract

Compute a subset of kernels.

- Double-cube kernel extraction [Rajski et al ‘90] (***fast-extract***)
  - single-cube double-literal kernels.
  - double-cube divisors.

Properties of fast extract

- Single- and double-cube divisors are considered concurrently.
- Handle divisor and complemented divisor simultaneously.
- Double-cube divisors are found using a pairwise comparison between cubes of the same Boolean function.
- The weight of each divisor is a function of the number of saved literals and its logic level.

Fast-extract Algorithm

1. Generate all 2-literal kernels and stores it's complemented form.
2. Generate and store all 2-cube kernels.
3. Choose the best divisor and extract it.
4. Update the set of divisors.
5. Iterate extraction of divisors until no more improvement.

Table of Contents

Differences

FX

- Divisors can be any function containing up to N literals - where N has the default value of 4, but can be defined by the user.
- **Can't** handle "degenerate" divisors.
- Common divisors are found by enumerating cube pairs.
- **Doesn't keep** track of the relation between divisors and cube pairs.

FXCH

- Restricts divisor to a small set of functions (1, NAND, XOR, MUX).
- **Can** handle "degenerate" divisors.
- Common divisors are found with the help of a sub-cube hash table.
- **Keep** track of the relation between divisors and cube pairs.
- Uses a multiple-output representation for cubes.

Degenerate divisors

There is a set of divisors that can deteriorate the outcome of extraction if not properly handled:

- The constant-1 divisor: $(\overline{x}_i + x_i)$ (means SCC)
- Handling $x_i + \overline{x}_i x_k$, $x_i \overline{x}_k + x_k$ and $x_i + x_k$ as three different divisors, while in fact they are the same divisor $x_i + x_k$

Creating single-cube divisors

Creating single-cube divisors

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

Both FX and FXCH use the same technique to create single cube divisors

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Creating single-cube divisors

$$f = x_1x_2e_1e_2 + x_3e_1e_2 + x_1x_2t_1t_2t_3 + x_3t_1t_2t_3 + \overline{x}_1\overline{x}_3v_1v_2 + \overline{x}_2\overline{x}_3v_1v_2$$

Both FX and FXCH use the same technique to create single cube divisors

1. They select one cube at a time.

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Creating single-cube divisors

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Both FX and FXCH use the same technique to create single cube divisors

1. They select one cube at a time.
2. For the selected cube, literals pairs are enumerated. For each pair of literals a divisor is created.

Divisors:

- $x_1 x_2$

Creating single-cube divisors

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Both FX and FXCH use the same technique to create single cube divisors

1. They select one cube at a time.
2. For the selected cube, literals pairs are enumerated. For each pair of literals a divisor is created.

Divisors:

- $x_1 x_2$
- $x_1 e_1$

Creating single-cube divisors

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Both FX and FXCH use the same technique to create single cube divisors

1. They select one cube at a time.
2. For the selected cube, literals pairs are enumerated. For each pair of literals a divisor is created.

Divisors:

- $x_1 x_2$
- $x_1 e_1$
- $x_1 e_2$

Creating single-cube divisors

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Both FX and FXCH use the same technique to create single cube divisors

1. They select one cube at a time.
2. For the selected cube, literals pairs are enumerated. For each pair of literals a divisor is created.

Divisors:

- $x_1 x_2$
- $x_1 e_1$
- $x_1 e_2$
- $x_2 e_1$

Creating single-cube divisors

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Both FX and FXCH use the same technique to create single cube divisors

1. They select one cube at a time.
2. For the selected cube, literals pairs are enumerated. For each pair of literals a divisor is created.

Divisors:

- $x_1 x_2$
- $x_1 e_1$
- $x_1 e_2$
- $x_2 e_1$
- $x_2 e_2$

Creating single-cube divisors

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Both FX and FXCH use the same technique to create single cube divisors

1. They select one cube at a time.
2. For the selected cube, literals pairs are enumerated. For each pair of literals a divisor is created.

Divisors:

- $x_1 x_2$
- $x_1 e_1$
- $x_1 e_2$
- $x_2 e_1$
- $x_2 e_2$
- $e_1 e_2$

Creating single-cube divisors

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Both FX and FXCH use the same technique to create single cube divisors

1. They select one cube at a time.
2. For the selected cube, literals pairs are enumerated. For each pair of literals a divisor is created.
3. Each created divisor is added to the divisors hash table and has its weight calculated.

Divisors:

- $x_1 x_2$
- $x_1 e_1$
- $x_1 e_2$
- $x_2 e_1$
- $x_2 e_2$
- $e_1 e_2$

Actually, we store the negation of the found single-cube divisors, ie:

$$\overline{x_1 x_2} = \overline{x}_1 + \overline{x}_2$$

Creating double-cube divisors

The FX way

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes. This behaviour is $O(n^2)$, where n is the number of cubes.

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.
- For each cube pair: literals are compared.

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.
- For each cube pair: literals are compared.

Divisor:

$$\frac{c_i}{c_i \cap c_j} + \frac{c_j}{c_i \cap c_j}$$

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.
- For each cube pair: literals are compared.

Divisor:

$$x_1 x_2 + x_3$$

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.
- For each cube pair: literals are compared.

Divisor:

$$x_1 x_2 + x_3$$

$$\overline{x}_1 + \overline{x}_2$$

Creating double-cube divisors - The FX way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- Find common divisors by pairwise comparison of cubes.
- For each cube pair: literals are compared.

Divisor:

$$x_1 x_2 + x_3$$
$$\overline{x}_1 + \overline{x}_2$$

Here is the reason why, when creating single cube divisors, we stored the negation of $x_1 x_2$. (Normalization)

FX - Result

$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$

The first picked divisor is $x_1 x_2 + x_3$ which lead to:

- $f = G e_1 e_2 + G t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$
- $G = x_1 x_2 + x_3$

FX - Result

$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$

The first picked divisor is $x_1 x_2 + x_3$ which lead to:

- $f = Ge_1 e_2 + Gt_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$
- $G = x_1 x_2 + x_3$

Followed by: $x_1 x_2$ $(\overline{x}_1 + \overline{x}_2)$

- $f = Ge_1 e_2 + Gt_1 t_2 t_3 + \overline{H} \overline{x}_3 v_1 v_2$
- $G = H + x_3$
- $H = x_1 x_2$

FX - Result

$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$

The first picked divisor is $x_1 x_2 + x_3$ which lead to:

- $f = Ge_1 e_2 + Gt_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$
- $G = x_1 x_2 + x_3$

Followed by: $x_1 x_2$ $(\overline{x}_1 + \overline{x}_2)$

- $f = Ge_1 e_2 + Gt_1 t_2 t_3 + \overline{H} \overline{x}_3 v_1 v_2$
- $G = H + x_3$
- $H = x_1 x_2$

Followed by: $\overline{H} \overline{x}_3$ $(H + x_3)$

- $f = Ge_1 e_2 + Gt_1 t_2 t_3 + \overline{K} v_1 v_2$
- $G = \overline{K}$
- $H = x_1 x_2$
- $K = \overline{H} \overline{x}_3$

Creating double-cube divisors

The FXCH way

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Find common divisors with the help of a sub-cube hash table.

- For each cube FXCH will:

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

- For each cube FXCH will:
  - Add the cube itself to a hash table.

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Cubes	
$x_1 x_2 e_1 e_2$	

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

Cubes	**Sub-cubes**
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Cubes	**Sub-cubes**
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$ $e_1 e_2$

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

Cubes	**Sub-cubes**
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$ $e_1 e_2$ $x_2 e_2$

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

Cubes	**Sub-cubes**			
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$	$e_1 e_2$	$x_2 e_2$	$x_2 e_1$

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Cubes	**Sub-cubes**			
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$	$e_1 e_2$	$x_2 e_2$	$x_2 e_1$
	$x_1 e_1 e_2$			

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

Cubes	**Sub-cubes**
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$ $e_1 e_2$ $x_2 e_2$ $x_2 e_1$
	$x_1 e_1 e_2$ $x_1 e_2$

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

Cubes	**Sub-cubes**			
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$	$e_1 e_2$	$x_2 e_2$	$x_2 e_1$
	$x_1 e_1 e_2$	$x_1 e_2$	$x_1 e_1$	

Creating double-cube divisors - The FXCH way

$$f = x_1x_2e_1e_2 + x_3e_1e_2 + x_1x_2t_1t_2t_3 + x_3t_1t_2t_3 + \overline{x}_1\overline{x}_3v_1v_2 + \overline{x}_2\overline{x}_3v_1v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

Cubes	**Sub-cubes**			
$x_1x_2e_1e_2$	$x_2e_1e_2$	e_1e_2	x_2e_2	x_2e_1
	$x_1e_1e_2$	x_1e_2	x_1e_1	
	$x_1x_2e_2$			

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Cubes	**Sub-cubes**			
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$	$e_1 e_2$	$x_2 e_2$	$x_2 e_1$
	$x_1 e_1 e_2$	$x_1 e_2$	$x_1 e_1$	
	$x_1 x_2 e_2$	$x_1 x_2$		

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

Cubes	**Sub-cubes**			
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$	$e_1 e_2$	$x_2 e_2$	$x_2 e_1$
	$x_1 e_1 e_2$	$x_1 e_2$	$x_1 e_1$	
	$x_1 x_2 e_2$	$x_1 x_2$		
	$x_1 x_2 e_1$			

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Cubes	**Sub-cubes**			
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$	$e_1 e_2$	$x_2 e_2$	$x_2 e_1$
$x_3 e_1 e_2$	$x_1 e_1 e_2$	$x_1 e_2$	$x_1 e_1$	
	$x_1 x_2 e_2$	$x_1 x_2$		
	$x_1 x_2 e_1$			

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

Cubes	**Sub-cubes**			
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$	$e_1 e_2$	$x_2 e_2$	$x_2 e_1$
$x_3 e_1 e_2$	$x_1 e_1 e_2$	$x_1 e_2$	$x_1 e_1$	
	$x_1 x_2 e_2$	$x_1 x_2$		
	$x_1 x_2 e_1$			

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.
  - Imagining a perfect hashing, a hit means that a common divisor exists.

Cubes	**Sub-cubes**			
$x_1 x_2 e_1 e_2$	$x_2 e_1 e_2$	$e_1 e_2$	$x_2 e_2$	$x_2 e_1$
$x_3 e_1 e_2$	$x_1 e_1 e_2$	$x_1 e_2$	$x_1 e_1$	
	$x_1 x_2 e_2$	$x_1 x_2$		
	$x_1 x_2 e_1$			

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
    - Add the cube itself to a hash table.
    - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.
    - Imagining a perfect hashing, a hit means that a common divisor exists.

Indeed, it found the divisor:

$$x_1 x_2 + x_1$$

Which is created using the removed literals of each cube.

Creating double-cube divisors - The FXCH way

$$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$$

x_1	x_2	e_1	e_2	
x_3	e_1	e_2		
x_1	x_2	t_1	t_1	t_3
x_3	t_1	t_2	t_3	
$\overline{x}_1$	$\overline{x}_3$	v_1	v_2	
$\overline{x}_2$	$\overline{x}_3$	v_1	v_2	

- For each cube FXCH will:
  - Add the cube itself to a hash table.
  - Generated all sub-cubes by removing 1 and 2 literals, and add them to the same hash table.
  - Imagining a perfect hashing, a hit means that a common divisor exists.

Indeed, it found the divisor:

$$x_1 x_2 + x_1$$

Which is created using the removed literals of each cube. **But FXCH limits divisors to a small set of functions (1, NAND, XOR, MUX)!**
So, this divisor is discarded.

FXCH - Result

$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$

The first picked divisor is $x_1 x_2$ $(\overline{x}_1 + \overline{x}_2)$ which lead to:

- $f = Ge_1 e_2 + x_3 e_1 e_2 + Gt_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{G} \overline{x}_3 v_1 v_2$
- $G = x_1 x_2$

FXCH - Result

$f = x_1 x_2 e_1 e_2 + x_3 e_1 e_2 + x_1 x_2 t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{x}_1 \overline{x}_3 v_1 v_2 + \overline{x}_2 \overline{x}_3 v_1 v_2$

The first picked divisor is $x_1 x_2$ $(\overline{x}_1 + \overline{x}_2)$ which lead to:

- $f = G e_1 e_2 + x_3 e_1 e_2 + G t_1 t_2 t_3 + x_3 t_1 t_2 t_3 + \overline{G x_3} v_1 v_2$
- $G = x_1 x_2$

Followed by: $\overline{G x_3}$ $(G + x_3)$

- $f = \overline{H} e_1 e_2 + \overline{H} t_1 t_2 t_3 + H v_1 v_2$
- $G = x_1 x_2$
- $H = \overline{G x_3}$

FXCH - Limiting divisor functions

The complete set of possible double-cube divisors with complements is summarized in Theorem 1 in [Rajski et al '92]. The set implies that using canonical basis NAND, XOR ($\oplus$), and MUX as divisor functions imposes a duality property in such divisors, meaning that the **complement of a divisor is also a divisor.**

FXCH - Limiting divisor functions

The complete set of possible double-cube divisors with complements is summarized in Theorem 1 in [Rajski et al '92]. The set implies that using canonical basis NAND, XOR ($\oplus$), and MUX as divisor functions imposes a duality property in such divisors, meaning that the **complement of a divisor is also a divisor.**

Observe that this limitation forbids factors of the form $x_1 x_2 + x_3$, because its complement function $\overline{x}_1 \overline{x}_3 + \overline{x}_2 \overline{x}_3$ is not a kernel, hence can not be used by algebraic extraction.

FXCH - Limiting divisor functions

The complete set of possible double-cube divisors with complements is summarized in Theorem 1 in [Rajski et al '92]. The set implies that using canonical basis NAND, XOR ($\oplus$), and MUX as divisor functions imposes a duality property in such divisors, meaning that the **complement of a divisor is also a divisor.**

Observe that this limitation forbids factors of the form $x_1x_2 + x_3$, because its complement function $\overline{x}_1\overline{x}_3 + \overline{x}_2\overline{x}_3$ is not a kernel, hence can not be used by algebraic extraction.

This means that when FX chooses to extract $x_1x_2 + x_3$, it does not account for its complement. This affects the relative balance of aggregated scores and leads to an inaccuracy of potential savings, which is likely to be more pronounced in larger problems.

FXCH - Cube Grouping

x_1	x_2	x_3	y_1	y_2
1	1	-	1	0
-	1	1	1	0
0	0	0	1	1
0	1	1	1	1
1	-	1	0	1
1	1	0	0	1

y_1	x_1	x_2	
y_1	x_2	x_3	
y_1	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$
y_1	$\overline{x_1}$	x_2	x_3
y_2	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$
y_2	$\overline{x_1}$	x_2	x_3
y_2	x_1	x_3	
y_2	x_1	x_2	$\overline{x_3}$

FXCH - Cube Grouping

x_1	x_2	x_3	y_1	y_2
1	1	-	1	0
-	1	1	1	0
0	0	0	1	1
0	1	1	1	1
1	-	1	0	1
1	1	0	0	1

y_1	x_1	x_2	
y_1	x_2	x_3	
$y_1 y_2$	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$
$y_1 y_2$	$\overline{x}_1$	x_2	x_3
y_2	x_1	x_3	
y_2	x_1	x_2	$\overline{x}_3$

Table of Contents

Results - Cube Grouping

Table: The impact of using cube grouping

	Run-time (s)		Memory (Mb)	
	w/ CG	w/o CG	w/ CG	w/o CG
37/143	2.95	14.47	113.36	673.14
38/67	1.16	2.75	58.64	265.11
128/43	1.90	4.73	105.48	430.69
128/53	1.63	3.89	102.21	421.89
128/55	2.03	6.44	104.95	523.19
128/69	2.72	14.64	106.84	556.56
128/94	4.56	29.38	120.44	1013.58
128/104	3.98	24.25	120.61	915.76
128/160	8.35	56.71	226.32	1882.85
ratios:	**0.19**	**1**	**0.16**	**1**

Results - EPFL Multiple-output PLA Behnchmakrs

Table: Logic synthesis results: comparison of *fxch*, *fx* and *jee*

	fxch in ABC				*fx* in ABC				*jee* factoring			
Design	t, sec	m, Mb	#nodes	#lvl	t, sec	m, Mb	#nodes	#lvl	t, sec	m, Mb	#nodes	#lvl
37/143	2.95	113	3835	22	18.18	11	4695	24	4.3	37	3587	24
38/67	1.16	59	3438	19	2.14	7	3727	18	2.0	25	3366	20
128/43	1.90	105	3051	18	2.43	9	3702	18	2.3	22	3191	18
128/53	1.63	102	2708	18	2.09	9	3261	19	2.0	23	2944	19
128/55	2.03	105	3079	18	2.46	10	3905	18	2.1	22	3069	20
128/69	2.72	107	3415	19	4.60	14	4295	20	2.7	28	3326	20
128/94	4.56	120	5140	21	9.50	20	6271	22	6.3	46	5266	24
128/104	3.98	121	4916	20	7.61	17	5853	21	5.7	44	4926	23
128/160	8.35	226	7358	23	21.02	30	8889	24	15.5	76	7268	24
ratios:	**0.42**	**8.33**	**0.83**	**0.97**	**1**	**1**	**1**	**1**	**0.61**	**2.54**	**0.83**	**1.04**

Results - Primes (Scalability)

Table: Results of the synthesis of the primality testing circuits.

	fxch in ABC				*fx* in ABC				*jee* factoring			
#inputs	t, sec	m, Mb	#nodes	#lvl	t, sec	m, Mb	#nodes	#lvl	t, sec	m, Mb	#nodes	#lvl
11	0.02	7	455	13	0.03	2	471	13	-	3	492	13
12	0.04	13	739	14	0.13	2	771	14	0.1	5	825	14
13	0.10	25	1355	15	0.53	2	1440	15	0.3	9	1419	15
14	0.25	50	2046	16	2.07	3	2401	16	1.4	20	2287	16
15	0.62	100	3670	17	7.99	5	4174	17	8.5	58	3989	17
16	1.55	202	6289	18	30.87	11	7448	18	41.2	151	6491	18
17	3.91	407	11413	19	129	23	11650	19	66.7	157	12096	19
18	12.97	827	17260	20	507	72	22158	20	167.8	169	18144	19
ratios:	**0.03**	**13.6**	**0.86**	**1**	**1**	**1**	**1**	**1**	**0.42**	**4.8**	**0.91**	**1**