Expressing Flexibility in Logic Synthesis by Boolean Relations

Anna Bernasconi
Università di Pisa, Italy

Overview

- Classic applications
  - Multilevel logic optimization
- Approximate Logic Synthesis
- Bounded-depth logic synthesis via Boolean relations
  - Bi-decomposed Circuits
  - Synthesis with critical signals: P-circuits

Multilevel logic optimization

Given a multilevel logic network, obtain an **equivalent** representation of the network, **optimal** w.r.t. a cost function involving area and delay

- *identifying subnetworks* to be optimized,
- deriving their *flexibility*
- and *replacing* such subnetworks by *simpler, optimized* ones

Single-output subnetworks

The flexibility for implementing the node's function can be represented by don't cares

Multilevel logic optimization

Given a multilevel logic network, obtain an **equivalent** representation of the network, **optimal** w.r.t. a cost function involving area and delay

- *identifying subnetworks* to be optimized,
- deriving their *flexibility*
- and *replacing* such subnetworks by *simpler, optimized* ones

Single-output subnetworks

The flexibility for implementing the node's function can be represented by don't cares

Multi-output subnetworks

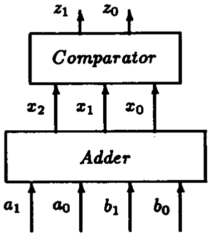
Don't cares are not sufficient for representing all the flexibility

> Don't care-based methods allow us to optimize only *one single-output subnetwork at a time*

Boolean relations describe all the flexibility

> *Boolean relations allow the simultaneous modification of all nodes of a subnetwork*

Minimization of a *two-bit adder* due to the filtering effect of a *comparator* [Brayton, Somenzi, 1989]



$z = 01 \Rightarrow a + b < 3$

$z = 00 \Rightarrow (a + b = 3) \vee (a + b = 4)$

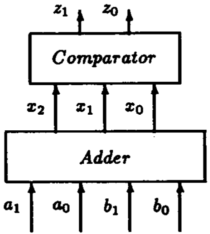
$z = 10 \Rightarrow a + b > 4.$

Input values can be partitioned into three equivalence classes:

Values less than 3: {000, 001, 010}
Values equal to 3 or 4: {011, 100}
Values greater than 4: {101, 110, 111}

The values in each class are not distinghished by the comparator

Minimization of a *two-bit adder* due to the filtering effect of a *comparator* [Brayton, Somenzi, 1989]



$z = 01 \Rightarrow a + b < 3$

$z = 00 \Rightarrow (a + b = 3) \vee (a + b = 4)$

$z = 10 \Rightarrow a + b > 4.$

Input values can be partitioned into three equivalence classes:

Values less than 3: {000, 001, 010}
Values equal to 3 or 4: {011, 100}
Values greater than 4: {101, 110, 111}

The values in each class are not distinghished by the comparator

We can change the output value of the adder, to any other value in the same equivalence class.

Boolean relation describing the flexible adder

$a_1\, a_0\, b_1\, b_0$	$x_2\, x_1\, x_0$
$\{0000, 0001, 0010, 0100, 1000, 0101\}$	$\{000, 001, 010\}$
$\{0011, 0110, 1001, 1010, 1100, 0111, 1101\}$	$\{011, 100\}$
$\{1011, 1110, 1111\}$	$\{101, 110, 111\}$

Minimization of a *two-bit adder* due to the filtering effect of a *comparator* [Brayton, Somenzi, 1989]

Minimization with don't cares

including the normal adder as an acceptable implementation

$a_1\ a_0\ b_1\ b_0$	$x_2\ x_1\ x_0$
1 1 - 0	0 1 1
- 1 1 0	0 1 1
1 0 - 1	0 1 1
- 0 1 1	0 1 1
- 1 1 1	1 0 0
1 1 - 1	1 0 0
1 1 1 -	0 1 0
1 - 1 -	1 0 0

Minimization of the Boolean relation

much simpler minimum solution

$a_1\ a_0\ b_1\ b_0$	$x_2\ x_1\ x_0$
0 - 1 -	0 1 0
1 - 0 -	0 1 0
1 - 1 -	1 0 0
- - - 1	0 0 1
- 1 - -	0 0 1

Approximate Logic Synthesis (ALS)

- Exploit error tolerance of applications to implement approximate designs with
  - smaller area
  - smaller delay
  - or lower energy consumption
- Modify some outputs of a function, so that the produced error is tolerable

Approximate Logic Synthesis (ALS)

- Exploit error tolerance of applications to implement approximate designs with
  - smaller area
  - smaller delay
  - or lower energy consumption
- Modify some outputs of a function, so that the produced error is tolerable

Error frequency

number of minterms on which an error occurs, as a fraction of the total number of minterms

Error magnitude

maximum amount by which the numerical value at the outputs of a function can deviate from the exact value

ALS and Boolean Relations

[Miao, Gerstlauer, Orshansky, 2013]:

- ALS under arbitrary *error magnitude* and *error frequency* constraints
- Two-level logic minimization algorithm, two-phase approach:

1. derive the solution of the problem constrained only by the *magnitude of errors*

2. the solution is iteratively refined to meet the original *error frequency constraint*

ALS and Boolean Relations

[Miao, Gerstlauer, Orshansky, 2013]:

- ALS under arbitrary *error magnitude* and *error frequency* constraints
- Two-level logic minimization algorithm, two-phase approach:

1. derive the solution of the problem constrained only by the *magnitude of errors* → expressed and solved using Boolean relations

2. the solution is iteratively refined to meet the original *error frequency constraint*

ALS constrained by Error Magnitude only

- Multi-output function $f : \{0,1\}^n \rightarrow \{0,1\}^k$
- M: constrain on the magnitude of possible errors

PROBLEM

Find $f' : \{0,1\}^n \rightarrow \{0,1\}^k$ of **minimal cost** s.t.

$$\forall x \in \{0,1\}^n \quad |f(x) - f'(x)| \leq M$$

ALS constrained by Error Magnitude only

- Multi-output function $f : \{0,1\}^n \rightarrow \{0,1\}^k$
- M: constrain on the magnitude of possible errors

Problem

Find $f' : \{0,1\}^n \rightarrow \{0,1\}^k$ of **minimal cost** s.t.

$$\forall x \in \{0,1\}^n \quad |f(x) - f'(x)| \leq M$$

- $\forall x \in \{0,1\}^n$, $e(x) =$ output error set for x

  additional values that the function can take while satisfying the error magnitude constraint

$$\mathcal{R}_{f'}(x) \in \{f(x) \cup e(x)\}$$

each input corresponds to more than one output: f' becomes a Boolean relation $\mathcal{R}_{f'}$

$\Rightarrow$ **minimize the Boolean relation $\mathcal{R}_{f'}$**, under a given metric (i.e., the number of literals in a SOP representation)

Example

ADDER

$x_1\ x_2$	$f(x_1, x_2)$
0 0	00
0 1	01
1 0	01
1 1	10

$$L(f) = 6$$

$$SOP(f^{(1)}) = x_1 x_2$$
$$SOP(f^{(2)}) = \overline{x}_1 x_2 + x_1 \overline{x}_2$$

Example

ADDER

$x_1\, x_2$	$f(x_1, x_2)$
0 0	00
0 1	01
1 0	01
1 1	10

$$L(f) = 6$$

$$SOP(f^{(1)}) = x_1 x_2$$
$$SOP(f^{(2)}) = \overline{x}_1 x_2 + x_1 \overline{x}_2$$

ADDER, $M = 1$

$x_1\, x_2$	$\mathcal{R}_{f'}(x_1, x_2)$
0 0	$\{00, 01\}$
0 1	$\{01, 00, 10\}$
1 0	$\{01, 00, 10\}$
1 1	$\{10, 01, 11\}$

Example

ADDER

$x_1\ x_2$	$f(x_1, x_2)$
0 0	00
0 1	01
1 0	01
1 1	10

$$L(f) = 6$$

$$SOP(f^{(1)}) = x_1 x_2$$
$$SOP(f^{(2)}) = \overline{x}_1 x_2 + x_1 \overline{x}_2$$

ADDER, $M = 1$

$x_1\ x_2$	$\mathcal{R}_{f'}(x_1, x_2)$
0 0	$\{00, 01\}$
0 1	$\{01, 00, 10\}$
1 0	$\{01, 00, 10\}$
1 1	$\{10, 01, 11\}$

$$L(f') = 0$$

$$SOP(f^{(1)}) = 0$$
$$SOP(f^{(2)}) = 1$$

Example

ADDER

$x_1\ x_2$	$f(x_1, x_2)$
0 0	00
0 1	01
1 0	01
1 1	10

$L(f) = 6$

$SOP(f^{(1)}) = x_1 x_2$
$SOP(f^{(2)}) = \overline{x}_1 x_2 + x_1 \overline{x}_2$

ADDER, $M = 1$

$x_1\ x_2$	$\mathcal{R}_{f'}(x_1, x_2)$	
0 0	$\{00, 01\}$	X
0 1	$\{01, 00, 10\}$	
1 0	$\{01, 00, 10\}$	
1 1	$\{10, 01, 11\}$	X

$L(f') = 0$

$SOP(f^{(1)}) = 0$
$SOP(f^{(2)}) = 1$

ERROR FREQUENCY: 50 %

Frequency constrained ALS algorithm

- The BR solution may not satisfy the **constrain on error frequency**
- Iterative and greedy algorithm for **systematically corrects the wrong outputs** (leading to the smallest cost increase) **until the error frequency constraint is met**

Frequency constrained ALS algorithm

- The BR solution may not satisfy the **constrain on error frequency**
- Iterative and greedy algorithm for **systematically corrects the wrong outputs** (leading to the smallest cost increase) **until the error frequency constraint is met**

Error frequency: 25 %

$x_1\ x_2$	$f(x_1, x_2)$	$\mathcal{R}_{f'}(x_1, x_2)$	
0 0	00	$\{00, 01\}$	X
0 1	01	$\{01, 00, 10\}$	
1 0	01	$\{01, 00, 10\}$	
1 1	10	$\{10, 01, 11\}$	X

Frequency constrained ALS algorithm

- The BR solution may not satisfy the **constrain on error frequency**
- Iterative and greedy algorithm for **systematically corrects the wrong outputs** (leading to the smallest cost increase) **until the error frequency constraint is met**

Error frequency: 25 %

$x_1\, x_2$	$f(x_1, x_2)$	$\mathcal{R}_{f'}(x_1, x_2)$
0 0	00	$\{00, 01\}$
0 1	01	$\{01, 00, 10\}$
1 0	01	$\{01, 00, 10\}$
1 1	10	$\{10, 01, 11\}$ X

$$L(f') = 2$$

$$SOP(f^{(1)}) = 0$$
$$SOP(f^{(2)}) = x_1 + x_2$$

Bounded-depth logic synthesis via Boolean relations: Bi-Decomposition (joint work with R. K. Brayton, V.Ciriani, G. Trucco, and T. Villa)

$f : \{0,1\}^n \to \{0,1,-\}$

$f = (f_{on}, f_{dc}, f_{off})$ can be covered with

→ a SOP derived by the on-set (+ some dc-points)

→ a POS resulting from the complement of a SOP for the off-set (+ some dc-points)

⋆ *Which one gives the best cover, the SOP or the POS form?*

⋆ *Can we study a form that is part in SOP and part in POS form, and is better than both?*

Bounded-depth logic synthesis via Boolean relations: Bi-Decomposition (joint work with R. K. Brayton, V.Ciriani, G. Trucco, and T. Villa)

$f : \{0,1\}^n \rightarrow \{0,1,-\}$

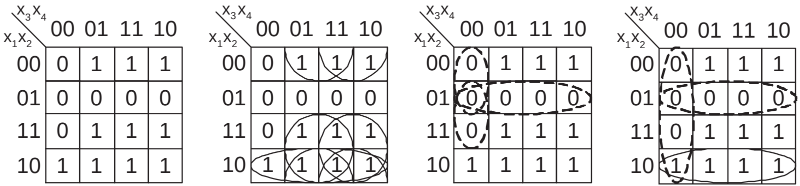
$f = (f_{on}, f_{dc}, f_{off})$ can be covered with

- $\rightarrow$ a SOP derived by the on-set (+ some dc-points)
- $\rightarrow$ a POS resulting from the complement of a SOP for the off-set (+ some dc-points)
  - ⋆ *Which one gives the best cover, the SOP or the POS form?*
  - ⋆ *Can we study a form that is part in SOP and part in POS form, and is better than both?*

We propose a **bi-decomposed form** that is part in SOP form and part in POS

$$f_B = \overline{f_0} \ \text{op} \ f_1$$

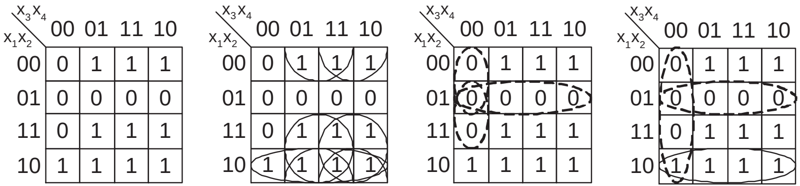
Bounded-depth logic synthesis via Boolean relations: Bi-Decomposition



(a) f (b) SOP for f (c) POS for f (d) Bi-cond. form for f

- $f_{SOP} = f_1^{SOP} = x_1\overline{x}_2 + x_1x_3 + x_1x_4 + \overline{x}_2x_3 + \overline{x}_2x_4$ 10 literals
- $f_{POS} = \overline{f}_0^{POS} = (x_1 + \overline{x}_2)(x_1 + x_3 + x_4)(\overline{x}_2 + x_3 + x_4)$ 8 literals

Bounded-depth logic synthesis via Boolean relations: Bi-Decomposition



(a) f (b) SOP for f (c) POS for f (d) Bi-cond. form for f

- $f_{SOP} = f_1^{SOP} = x_1\overline{x}_2 + x_1x_3 + x_1x_4 + \overline{x}_2x_3 + \overline{x}_2x_4$ 10 literals
- $f_{POS} = \overline{f}_0^{POS} = (x_1 + \overline{x}_2)(x_1 + x_3 + x_4)(\overline{x}_2 + x_3 + x_4)$ 8 literals
- $f_B = \overline{f}_0 + f_1 = ((x_1 + \overline{x}_2)(x_3 + x_4)) + x_1\overline{x}_2$ 6 literals

1000 is in the the OFF set of $\overline{f}_0$ and in the ON set of f_1
thus is in the ON set of the $\overline{f}_0 + f_1$

Synthesis of Bi-decomposed Circuits and Boolean relations

$f : \{0,1\}^n \to \{0,1,-\}$, $f = u \; \mathbf{op} \; v$ $u \leftarrow \overline{f_0},\; v \leftarrow f_1$

Inputs of u and v: the same as the inputs of f: $x_1, \ldots, x_n$
Output: is the output that **op** takes on $u(x_1, \ldots, x_n)$ and $v(x_1, \ldots, x_n)$

Synthesis of Bi-decomposed Circuits and Boolean relations

$f : \{0,1\}^n \rightarrow \{0,1,-\}$, $f = u \ \mathbf{op} \ v$ $u \leftarrow \overline{f_0},\ v \leftarrow f_1$

Inputs of u and v: the same as the inputs of f: $x_1, \ldots, x_n$
Output: is the output that **op** takes on $u(x_1, \ldots, x_n)$ and $v(x_1, \ldots, x_n)$

AND GROUP

$u\,v$	AND
$\overline{u}\,v$	$\not\Leftarrow$
$u\,\overline{v}$	$\not\Rightarrow$
$\overline{u}\,\overline{v}$	NOR

OR GROUP

$u + v$	OR
$\overline{u} + v$	$\Rightarrow$
$u + \overline{v}$	$\Leftarrow$
$\overline{u} + \overline{v}$	NAND

XOR GROUP

$u \oplus v$	XOR
$u \ \overline{\oplus}\ v$	XNOR

Synthesis of Bi-decomposed Circuits and Boolean relations

$f : \{0,1\}^n \rightarrow \{0,1,-\}$, $\quad$ $f = u \text{ } \mathbf{op} \text{ } v$ $\quad$ $u \leftarrow \overline{f_0},\ v \leftarrow f_1$

Inputs of u and v: the same as the inputs of f: $x_1, \ldots, x_n$
Output: is the output that **op** takes on $u(x_1, \ldots, x_n)$ and $v(x_1, \ldots, x_n)$

AND GROUP

$u\,v$	AND
$\overline{u}\,v$	$\not\Leftarrow$
$u\,\overline{v}$	$\not\Rightarrow$
$\overline{u}\,\overline{v}$	NOR

OR GROUP

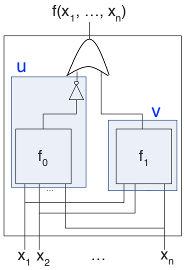
$u + v$	OR
$\overline{u} + v$	$\Rightarrow$
$u + \overline{v}$	$\Leftarrow$
$\overline{u} + \overline{v}$	NAND

XOR GROUP

$u \oplus v$	XOR
$u \overline{\oplus} v$	XNOR

- The two-input operator induces flexibility that cannot be expressed exactly by don't care conditions only: **a Boolean relation is required**
- For each binary **op**, we define $\mathcal{R}_{op} : \{0,1\}^n \rightarrow \{0,1\}^2$ s.t.
  - the set of functions compatible with $\mathcal{R}_{op}$ corresponds to the set of pairs (u, v) occurring in all bi-decomposed circuit implementations of f w.r.t. op.
  - an **optimal solution of $\mathcal{R}_{op}$** is an **optimal bi-decomposed circuit** for f

Construction of $\mathcal{R}_{OR}$



- $\forall x \in f_{on}$, x must be associated to one of the three output values on which $u + v$ evaluates to 1

$$\mathcal{R}_{OR}(x) = \{01, 10, 11\} = \{1-, -1\}$$

- $\forall x \in f_{off}$, x must be associated to the output 00 on which $u + v$ evaluates to 0

$$\mathcal{R}_{OR}(x) = 00$$

- $\forall x \in f_{dc}$, x can be associated to any output

$$\mathcal{R}_{OR}(x) = \{--\}$$

Construction of $\mathcal{R}_{op}$

AND table

	$\mathcal{R}_{AND}$	$\mathcal{R}_{\not\Leftarrow}$	$\mathcal{R}_{NOR}$	$\mathcal{R}_{\not\Rightarrow}$
$x \in f_{on}$	$\{11\}$	$\{01\}$	$\{00\}$	$\{10\}$
$x \in f_{off}$	$\{0-, -0\}$	$\{1-, -0\}$	$\{1-, -1\}$	$\{0-, -1\}$
$x \in f_{dc}$	$\{--\}$	$\{--\}$	$\{--\}$	$\{--\}$

OR table

	$\mathcal{R}_{OR}$	$\mathcal{R}_{\Rightarrow}$	$\mathcal{R}_{NAND}$	$\mathcal{R}_{\Leftarrow}$
$x \in f_{on}$	$\{1-, -1\}$	$\{0-, -1\}$	$\{0-, -0\}$	$\{1-, -0\}$
$x \in f_{off}$	$\{00\}$	$\{10\}$	$\{11\}$	$\{01\}$
$x \in f_{dc}$	$\{--\}$	$\{--\}$	$\{--\}$	$\{--\}$

XOR table

	$\mathcal{R}_{XNOR}$	$\mathcal{R}_{XOR}$
$x \in f_{on}$	$\{00, 11\}$	$\{01, 10\}$
$x \in f_{off}$	$\{01, 10\}$	$\{00, 11\}$
$x \in f_{dc}$	$\{--\}$	$\{--\}$

Construction of $\mathcal{R}_{op}$

The three tables are distinguished by whether

- the **offset** is partitioned **(AND group)**
- the **onset** is partitioned **(OR group)**
- or **both** are partitioned **(XOR group)**

How this partitioning is done is task of the *Boolean relation minimizer*

Construction of $\mathcal{R}_{op}$

The three tables are distinguished by whether

- the **offset** is partitioned **(AND group)**
- the **onset** is partitioned **(OR group)**
- or **both** are partitioned **(XOR group)**

How this partitioning is done is task of the *Boolean relation minimizer*

Bi-decomposed circuit minimization problem

$\Longleftrightarrow$

problem of finding an optimal implementation of $\mathcal{R}_{op}$

Good gains in a majority of benchmarks against affordable increases in synthesis time

Synthesis with critical signals: P-circuits
(joint work with V. Ciriani, G. Trucco, and T. Villa)

Scenario

- Logic synthesis in presence of critical signals that should be moved toward the output
- signals with high switching activity
  $\rightarrow$ for decreasing power consumption
- late arriving signals
  $\rightarrow$ for decreasing circuit delay

Synthesis with critical signals: P-circuits
(joint work with V. Ciriani, G. Trucco, and T. Villa)

Scenario

- Logic synthesis in presence of critical signals that should be moved toward the output
- signals with high switching activity
  $\rightarrow$ for decreasing power consumption
- late arriving signals
  $\rightarrow$ for decreasing circuit delay

Problem

Restructure (or synthesize) a circuit in order to move critical signals near to the output (decreasing the cone of influence)

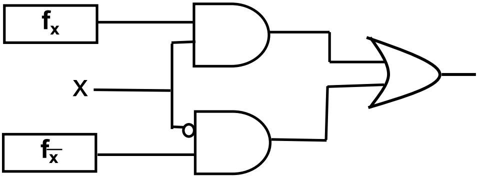
- **minimizing** the circuit **area**
- keeping the number of levels **bounded**
- performing an **efficient** minimization

Simple solution: Shannon decomposition

- x is the critical signal

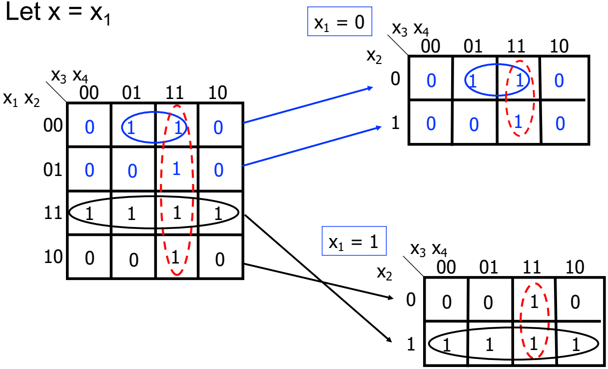
$$f = \overline{x}\, f_{|x=0} + x\, f_{|x=1}$$

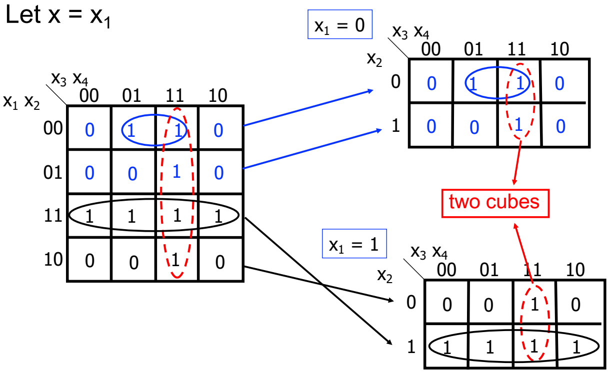
- the cofactors $f_{|x=0}$ and $f_{|x=1}$ do not depend on x
- x is near to the output



Problem of Shannon approach

Let $x = x_1$





It is not a compact representation

Decomposition with intersection

- try not to split the cubes
- let the critical signal near to the output

IDEA

- the cubes that **do not depend on x** and cross the two sets are not projected
- how to identify these cubes?

Decomposition with intersection

- try not to split the cubes
- let the critical signal near to the output

Idea

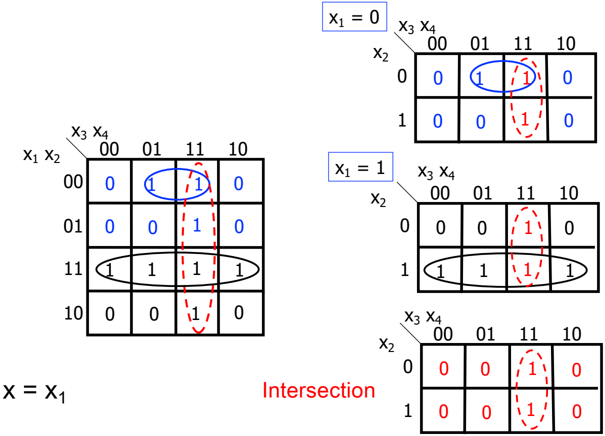
- the cubes that **do not depend on x** and cross the two sets are not projected
- how to identify these cubes?

  They are in the **intersection between the two cofactors**

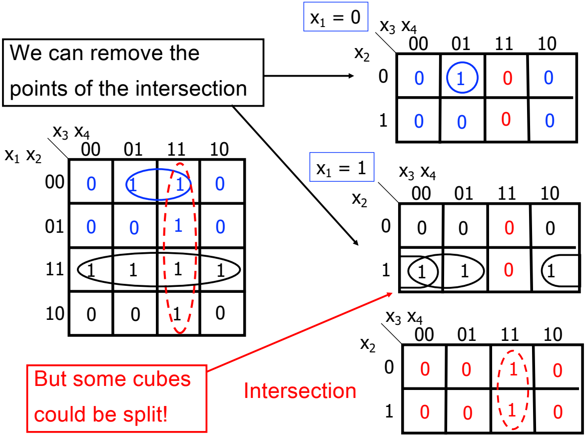
$$I = f_{|x_i=0} \cap f_{|x_i=1}$$

- keep I **unprojected**, and project only the minterms in $f_{|x_1=0} \setminus I$ and $f_{|x_i=1} \setminus I$

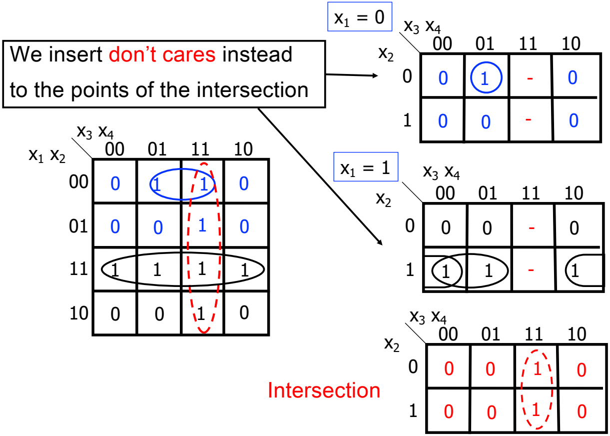
Example


x3 x4
x1 x2
00 01 11 10
00 0 1 1 0
01 0 0 1 0
11 1 1 1 1
10 0 0 1 0
x = x1
x1 = 0
x3 x4
x2
00 01 11 10
0 0 1 1 0
1 0 0 1 0
x1 = 1
x3 x4
x2
00 01 11 10
0 0 0 1 0
1 1 1 1 1
Intersection
x3 x4
x2
00 01 11 10
0 0 0 1 0
1 0 0 1 0

Example


We can remove the points of the intersection
But some cubes could be split!
Intersection
x1 = 0
x1 = 1
x3 x4
x1 x2
x2
00 01 11 10
00 0 1 1 0
01 0 0 1 0
11 1 1 1 1
10 0 0 1 0
0 0 1 0 0
1 0 0 0 0
0 0 0 0 0
1 1 1 0 1
0 0 0 1 0
1 0 0 1 0

Example


We insert don't cares instead
to the points of the intersection
x3 x4
x1 x2
00 01 11 10
00 0 1 1 0
01 0 0 1 0
11 1 1 1 1
10 0 0 1 0
x1 = 0
x3 x4
x2
00 01 11 10
0 0 1 - 0
1 0 0 - 0
x1 = 1
x3 x4
x2
00 01 11 10
0 0 0 - 0
1 1 1 - 1
x3 x4
x2
00 01 11 10
0 0 0 1 0
1 0 0 1 0
Intersection

Example

Decomposition with intersection

$x_1 x_2$ \ $x_3 x_4$	00	01	11	10
00	0	1	1	0
01	0	0	1	0
11	1	1	1	1
10	0	0	1	0

The cubes are not split

$x_1 = 0$

x_2 \ $x_3 x_4$	00	01	11	10
0	0	1	1	0
1	0	0	0	0

$x_1 = 1$

x_2 \ $x_3 x_4$	00	01	11	10
0	0	0	0	0
1	1	1	1	1

Intersection

x_2 \ $x_3 x_4$	00	01	11	10
0	0	0	1	0
1	0	0	1	0

P-circuits for completely specified functions

- if a **point is in** I and is useful for a better minimization of $f_{|x_i=0}$ and $f_{|x_i=1}$, **it can be kept both in the cofactors and in the intersection**
- if a **point is covered in both the projected cofactors**, it is **not necessary to cover it in** I (replaced by a don't care in I)

P-circuits for completely specified functions

- if a **point is in I** and is useful for a better minimization of $f_{|x_i=0}$ and $f_{|x_i=1}$, **it can be kept both in the cofactors and in the intersection**
- if a **point is covered in both the projected cofactors**, it is **not necessary to cover it in** I (replaced by a don't care in I)

P-circuit

A *P-circuit* of a completely specified function f is the circuit

$$P(f) = \overline{x}_i\, f^{=} + x_i\, f^{\neq} + f^{I}$$

1. $I = f_{|x_i=0} \cap f_{|x_i=1}$
2. $(f_{|x_i=0} \setminus I) \subseteq f^{=} \subseteq f_{|x_i=0}$
3. $(f_{|x_i=1} \setminus I) \subseteq f^{\neq} \subseteq f_{|x_i=1}$
4. $\emptyset \subseteq f^{I} \subseteq I$
5. $P(f) = f$

Minimization of P-circuits using Boolean Relation

Find the sets $f^{=}, f^{\neq}, f^{I}$ leading to a **P-circuit of minimal cost**

- $f : \{0,1\}^n \to \{0,1\}$ $\mathcal{R}_f : \{0,1\}^{n-1} \to \{0,1\}^3$
- **Input set for $\mathcal{R}_f$**: all input variables but the critical signal x_i
- **Output set for $\mathcal{R}_f$**: all triple of functions $f^{=}, f^{\neq}, f^{I}$ defining a P-circuit for f

Minimization of P-circuits using Boolean Relation

Find the sets $f^{=}, f^{\neq}, f^{I}$ leading to a **P-circuit of minimal cost**

- $f : \{0,1\}^n \to \{0,1\}$ $\mathcal{R}_f : \{0,1\}^{n-1} \to \{0,1\}^3$
- **Input set for $\mathcal{R}_f$**: all input variables but the critical signal x_i
- **Output set for $\mathcal{R}_f$**: all triple of functions $f^{=}, f^{\neq}, f^{I}$ defining a P-circuit for f

$x_1 \dots x_{i-1} x_{i+1} \dots x_n$	$\mathcal{R}_f = (f^{=}, f^{\neq}, f^{I})$
points in $f_{\vert x_i=0} \setminus I$	$\{100\}$
points in $f_{\vert x_i=1} \setminus I$	$\{010\}$
points in I	$\{--1, 11-\}$
all other points	$\{000\}$

Minimization of P-circuits using Boolean Relation

Find the sets $f^{=}, f^{\neq}, f^{I}$ leading to a **P-circuit of minimal cost**

- $f : \{0,1\}^n \rightarrow \{0,1\}$ $\quad\quad$ $\mathcal{R}_f : \{0,1\}^{n-1} \rightarrow \{0,1\}^3$
- **Input set for $\mathcal{R}_f$**: all input variables but the critical signal x_i
- **Output set for $\mathcal{R}_f$**: all triple of functions $f^{=}, f^{\neq}, f^{I}$ defining a P-circuit for f

$x_1 \dots x_{i-1} x_{i+1} \dots x_n$	$\mathcal{R}_f = (f^{=}, f^{\neq}, f^{I})$
points in $f_{\vert x_i=0} \setminus I$	$\{100\}$
points in $f_{\vert x_i=1} \setminus I$	$\{010\}$
points in I	$\{--1, 11-\}$
all other points	$\{000\}$

Theorem

P-circuit minimization for f

$\Longleftrightarrow$

minimization of the Boolean relation $\mathcal{R}_f$

P-circuits minimized with Boolean relations are more compact than P-circuits expressed and minimized as incompletely specified functions

Conclusions

- **Boolean relations can be extremely useful for modeling Boolean hard optimization problems**
  - → with Boolean relations we can model problems that cannot be completely described with incompletely specified functions
- **Problem**: scalability of the approach
  - Boolean relation minimization is a very hard problem
  - Boolean relation minimizers cannot handle relations with many outputs
- Boolean relations have been successfully used in logic synthesis to solve problems that can be cast as the minimization of *relations with a constant number of outputs* (2, 3)

THANK YOU