

Output Grouping Method Based on a Similarity of Boolean Functions

Petr Fišer, Pavel Kubalík, Hana Kubátová

*Czech Technical University in Prague
Department of Computer Science and Engineering*

Outline

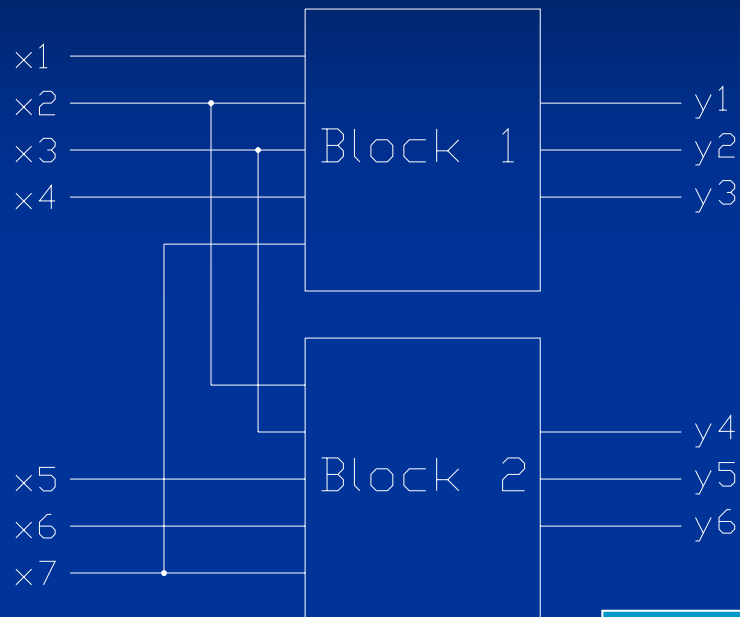
- Output Grouping
- Similarity-Based Method
- Experimental Results
- Conclusions

Output Grouping

Motivation

- There is a need to decompose the circuit into several stand-alone output-disjoint blocks
- Internal signals cannot be shared
- Inputs can be shared
- E.g., for PLAs, GALs, ...

Output Grouping



Motivation:

- Divide the circuit into stand-alone blocks
- Each block produces several outputs

Task:

- What outputs should be grouped together into one block, so that the overall logic is minimized?

Similarity-Based Output Grouping Method

Main principles:

- Based on joining functions (outputs), that are “somehow similar”
- Then there should be a big chance that the two functions will share a lot of logics

The Task:

- How to determine the measure of “similarity”?

Similarity-Based Output Grouping Method

Based on these straightforward observations:

- > Two equal functions are “very” similar
- > Two inverse functions are very similar too, since they could differ at most by one inverter in the final (multilevel) design



- > Two functions are similar, if a change of a value of one input variable induces a change of values of *both* the functions, or the values of *both* functions do not change. This should be checked for all possible input variable changes.

Method

“Two functions are similar, if a change of a value of one input variable induces a change of values of *both* the functions, or the values of *both* functions do not change. This should be checked for all possible input variable changes”

- Let us have a completely specified Boolean function given by a truth table (by minterms)
- For each minterm and each input variable change, observe the change of the two functions
- If values of both the functions are changed, or the values of both the functions unchanged, add 1 to the score
- The complexity can be halved, when only 0->1 changes are considered

Example

$$f = a + bc$$

$$f_2 = ab\bar{c} + \bar{a}c + a\bar{b}c$$

?

How much similar
are these two

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + ab\bar{c}$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm:** 000
- **Variable:** a (0->1)

f_1 : 0 -> 1
 f_2 : 0 -> 0

⇒ No score point

Score: 0

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + ab\bar{c}$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm:** 000
- **Variable:** b (0->1)

f_1 : 0 -> 0
 f_2 : 0 -> 0

⇒ + 1 score point

Score: 1

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + a\bar{b}c$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm:** 000
- **Variable:** c (0->1)

f_1 : 0 -> 0
 f_2 : 0 -> 1

⇒ No score point

Score: 1

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + a\bar{b}c$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm: 001**
- **Variable: a (0->1)**

$f_1: 0 \rightarrow 1$
 $f_2: 1 \rightarrow 1$

⇒ **No score point**

Score: 1

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + a\bar{b}c$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm:** 001
- **Variable:** b (0->1)

f_1 : 0 -> 1
 f_2 : 1 -> 1

⇒ No score point

Score: 1

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + a\bar{b}c$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm:** 010
- **Variable:** a (0->1)

f_1 : 0 -> 1
 f_2 : 0 -> 1

⇒ +1 score point

Score: 2

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + a\bar{b}c$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm: 010**
- **Variable: c (0->1)**

$f_1: 0 \rightarrow 1$
 $f_2: 0 \rightarrow 1$

$\Rightarrow +1$ score point

Score: 3

Example

$$f_1 = a + bc$$

$$f_2 = abc + \bar{a}c + a\bar{b}c$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm: 011**
- **Variable: a (0->1)**

$f_1: 1 \rightarrow 1$
 $f_2: 1 \rightarrow 0$

\Rightarrow No score point

Score: 3

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + a\bar{b}c$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm:** 100
- **Variable:** b (0->1)

f_1 : 1 -> 1
 f_2 : 0 -> 1

⇒ No score point

Score: 3

Example

$$f_1 = a + bc$$

$$f_2 = abc + \bar{a}c + ab\bar{c}$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm: 100**
- **Variable: c (0->1)**

f_1 : 1 -> 1
 f_2 : 0 -> 1

⇒ No score point

Score: 3

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + a\bar{b}c$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm:** 101
- **Variable:** b (0->1)

f_1 : 1 -> 1
 f_2 : 1 -> 0

⇒ No score point

Score: 3

Example

$$f_1 = a + bc$$

$$f_2 = abc\bar{c} + \bar{a}c + a\bar{b}c$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

- **Minterm: 110**
- **Variable: c (0->1)**

$f_1: 1 \rightarrow 1$
 $f_2: 1 \rightarrow 0$

⇒ **No score point**

Score: 3

Example

$$f_1 = a + bc$$

$$f_2 = abc + \bar{a}c + ab\bar{c}$$

abc	f_1	f_2
000	0	0
001	0	1
010	0	0
011	1	1
100	1	0
101	1	1
110	1	1
111	1	0

Score = 3

Method

Complexity

- $O(n \cdot 2^n)$
- Not that good...

Generalization of the Method

OK, we are counting differences and indifferences of the function value, when the input variable value is changed

- Boolean differential calculus does exactly the same work!

Generalization of the Method

“Two functions are similar, if a change of a value of one input variable induces a change of values of *both* the functions, or the values of *both* functions do not change. This should be checked for all possible input variable changes”

$$\frac{\partial f(x_0, \dots, x_n)}{\partial x_i} = f(x_0, \dots, x_i = 1, \dots, x_n) \oplus f(x_0, \dots, x_i = 0, \dots, x_n)$$

Generalization of the Method

“Two functions are similar, if a change of a value of one input variable induces a change of values of *both* the functions, **or the values of *both* functions do not change.** This should be checked for all possible input variable changes”

$$\frac{\bar{\partial}f(x_0, \dots, x_n)}{\bar{\partial}x_i} = f(x_0, \dots, x_i = 1, \dots, x_n) \Leftrightarrow f(x_0, \dots, x_i = 0, \dots, x_n)$$

Generalization of the Method

- Thus, we compute Boolean differences wrt. all input variables for the two functions
- Then their product is computed. This represents cubes, where the change of an input variable induces a change of values of both functions
- Then we compute Boolean indifferences wrt. all input variables for the two functions
- Then their product is computed. This represents cubes, where the change of an input variable induces no change of values of both functions
- The sizes of both these products are computed and summed together

$$s = \sum_{i=0}^{n-1} \left(\left| \frac{\partial f_1(x_0, \dots, x_n)}{\partial x_i} \cdot \frac{\partial f_2(x_0, \dots, x_n)}{\partial x_i} \right| + \left| \frac{\bar{\partial} f_1(x_0, \dots, x_n)}{\bar{\partial} x_i} \cdot \frac{\bar{\partial} f_2(x_0, \dots, x_n)}{\bar{\partial} x_i} \right| \right)$$

Generalization of the Method

Advantages

- Since Boolean algebra is involved, any algebraic representation of functions can be used
- Computing the product cubes takes linear time with the number of inputs

Disadvantage

- Well, computing their size is not that easy. Exponential time, in fact.

But it's not that bad in practice

Example

$$f_1 = a + bc$$

$$f_2 = abc + \bar{a}c + ab\bar{c}$$

Boolean differences:

$$\frac{\partial f_1}{\partial a} = (1 + bc) \oplus (bc) = \bar{b} + \bar{c}$$

$$\frac{\partial f_2}{\partial a} = (b\bar{c} + \bar{b}c) \oplus c = b$$

$$\frac{\partial f_1}{\partial a} \cdot \frac{\partial f_2}{\partial a} = (\bar{b} + \bar{c})b = b\bar{c}$$

$$|b\bar{c}| = 2$$

$$\frac{\partial f_1}{\partial b} = (a + c) \oplus a = \bar{a}c$$

$$\frac{\partial f_2}{\partial b} = (a\bar{c} + \bar{a}c) \oplus (\bar{a}c + ac) = a$$

$$\frac{\partial f_1}{\partial b} \cdot \frac{\partial f_2}{\partial b} = 0$$

$$|0| = 0$$

$$\frac{\partial f_1}{\partial c} = (a + b) \oplus a = \bar{a}b$$

$$\frac{\partial f_2}{\partial c} = (\bar{a} + ab) \oplus ab = 1$$

$$\frac{\partial f_1}{\partial c} \cdot \frac{\partial f_2}{\partial c} = \bar{a}b$$

$$|\bar{a}b| = 2$$

Total size = 4

Example

$$f_1 = a + bc$$

$$f_2 = abc + \bar{a}c + ab\bar{c}$$

Boolean indifference:

$$\frac{\bar{\partial}f_1}{\partial a} = (1 + bc) \Leftrightarrow (bc) = bc$$

$$\frac{\bar{\partial}f_2}{\partial a} = (b\bar{c} + \bar{b}c) \Leftrightarrow c = \bar{b}$$

$$\frac{\bar{\partial}f_1}{\partial a} \cdot \frac{\bar{\partial}f_2}{\partial a} = (bc) \cdot \bar{b} = 0$$

$$|0| = 0$$

$$\frac{\bar{\partial}f_1}{\partial b} = (a + c) \Leftrightarrow a = a + \bar{c}$$

$$\frac{\bar{\partial}f_2}{\partial b} = (a\bar{c} + \bar{a}c) \Leftrightarrow (\bar{a}c + ac) = \bar{a}$$

$$\frac{\bar{\partial}f_1}{\partial b} \cdot \frac{\bar{\partial}f_2}{\partial b} = \bar{a}\bar{c}$$

$$|\bar{a}\bar{c}| = 2$$

$$\frac{\bar{\partial}f_1}{\partial c} = (a + b) \Leftrightarrow a = a + \bar{b}$$

$$\frac{\bar{\partial}f_2}{\partial c} = (\bar{a} + ab) \Leftrightarrow ab = 0$$

$$\frac{\bar{\partial}f_1}{\partial c} \cdot \frac{\bar{\partial}f_2}{\partial c} = 0$$

$$|0| = 0$$

Total size = 2

Example

$$f_1 = a + bc$$

$$f_2 = abc + \bar{a}c + a\bar{b}c$$

$$\text{Total score} = 4 + 2 = 6$$

Recall that the truth-table based result was 3.

But without considering 1→0 changes, i.e., 6 in fact.

Scoring Matrix

When score for all function pairs is computed, the scoring matrix is constructed

- Symmetric matrix of dimensions (m, m) , where m is the number of output variables
- The value in a cell $[i, j]$ represents the scoring function value for outputs i and j
- The multi-output function's outputs are grouped together according the scoring matrix values

Scoring Matrix

Algorithm:

- Assign the first output variable to the first block. Since there is no relationship between outputs and blocks yet, it can be freely done.
- Find the maximum scoring matrix value, corresponding to outputs i and j . These outputs should be grouped together, since their “similarity value” is the highest one
- If one of these outputs is already assigned to a block, append the second one, if possible (maximum number of block’s outputs is not exceeded)
- If none of them is assigned, try to find an empty block and assign both outputs to this block
- If no free block is available, try to put them both into some block.
- If there is not enough place to put both the outputs into one block, assign them randomly

Experimental Results (1)

Decomposition of MCNC benchmark circuits

- First, the benchmark circuit has been minimized by BOOM. This experiment has been done to estimate the circuit size when no partitioning is used.
- Then we have divided the circuit into several blocks (b), while all the output variables were assigned to blocks *purely at random*. Then the circuit has been minimized by BOOM. 100 experiments were performed and an average value was taken, to ensure good statistical values.
- Finally the our method was used. We have made an experiment similar to the previously described one, but the output variables were assigned to the blocks using the proposed method.

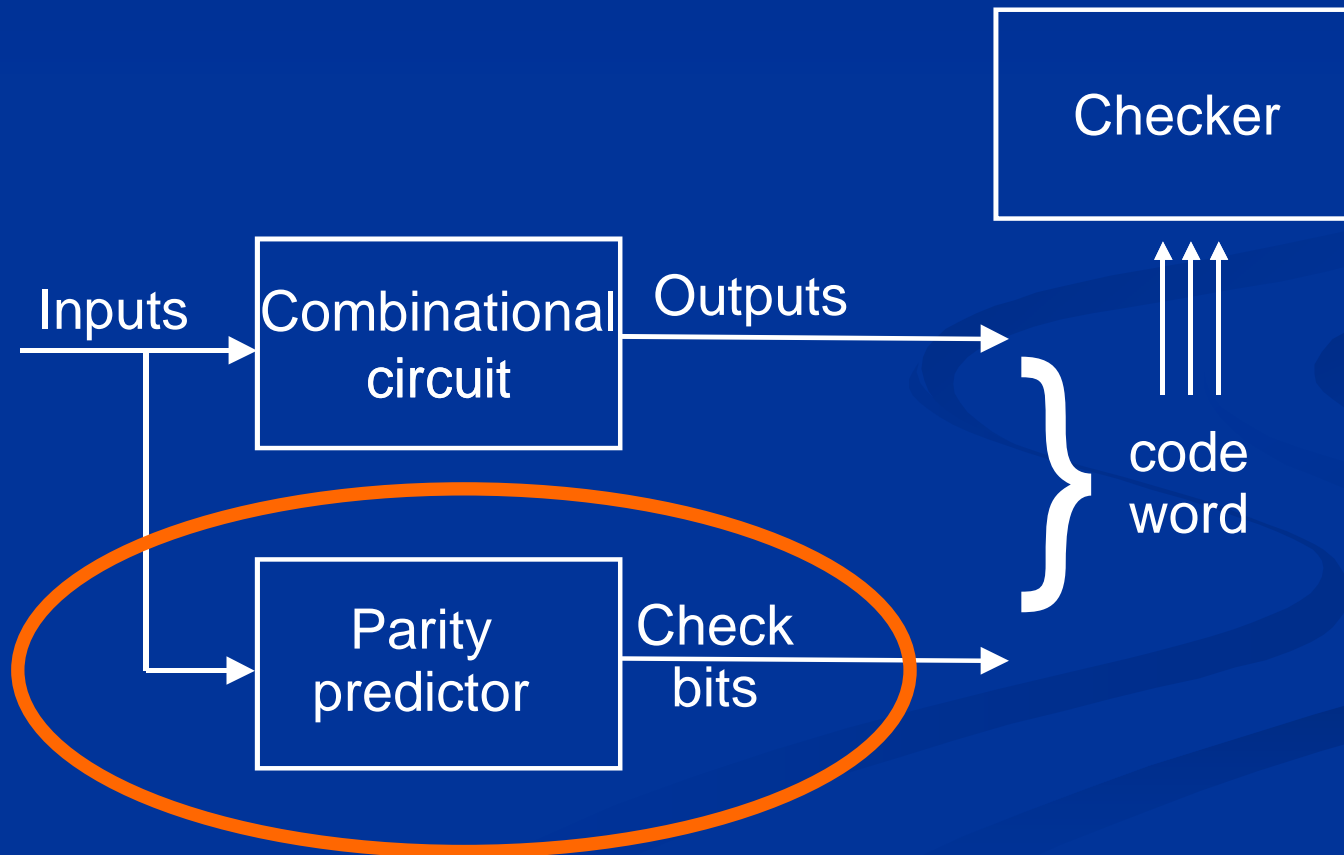
Experimental Results (1)

Decomposition of MCNC benchmark circuits

			No decomp.		Random output grouping	Similarity-based output grouping	
<i>bench</i>	<i>i</i>	<i>o</i>	<i>GEs</i>	<i>blocks</i>	<i>GEs</i>	<i>GEs</i>	<i>impr.</i>
al2	16	47	206.5	5	244.0	218.0	10.7%
amd	14	24	334.5	3	460.0	429.0	6.7%
b2	16	17	989.5	4	2018.5	1807.5	10.5%
b7	8	31	81.0	4	105.0	88.5	15.7%
b11	8	31	81.0	4	105.5	87.5	17%
br1	12	8	130.0	3	215.0	186.5	13%
dk17	10	11	70.5	3	84.0	72.5	13.7%
exps	8	38	910.0	4	1473.0	1256.0	14.7%
luc	8	27	162.5	3	244.0	228.0	6.6%

Experimental Results (2)

Application to on-line BIST



Experimental Results (2)

Application to on-line BIST

- The circuit outputs are gradually XORed (by 2), until the required number of parity bits is obtained
- Synthesis process is conducted after every output XORing
- Scoring matrix is recomputed in each step (number of outputs is decreased by 1)
- The choice of the outputs to be XORed is essential, to reduce the overall logics

Experimental Results (2)

Application to on-line BIST

Main ideas, similar to Similarity-based grouping:

- When two equal functions XORed, the result will be '0' for all minterms. If the values of two functions will differ in a couple of minterms only, there will be only several '1' values in the resulting XORed function. That's good for minimization.
- Two inverse functions, when XORed, yield a '1' value for all minterms. If the values of two functions are inverse but a few minterms, there will be only few '0' values in the result. That's good for minimization.
- And, consequently, if two functions are "similar", there is a big probability that they will share a lot of logic in the implemented design.

Experimental Results (2)

<i>Circuit</i>	<i>Random [GEs]</i>	<i>Similarity [GEs]</i>	<i>Red.</i>
alu1	967	156	83.9 %
apla	128	76	40.6 %
b11	36	21	41.7 %
br1	80	68	15 %
alu2	418	40	90.4 %
alu3	433	320	26.1 %
s1488	364	241	33.8 %
s386	87	73	16.1 %

Conclusions

- A new circuit decomposition and output grouping method based on a similarity of functions is proposed
- Its generalization based on Boolean difference is shown
- Experimental results confirm its efficiency