Kungl Tekniska Högskolan

Inte

Master Pr

in System-on-Chip Design

L7a: Three-level optimization

Reading material

- **TOP: An algorithm for three-level combinational logic optimization**, E. Dubrova, P. Ellervee, D.M. Miller, J.C. Muzio, A.J. Sullivan, *IEE Proceedings - Circuits, Devices and Systems*, vol. 5, no. 4, pp. 307-314, August 2004.

Three-level synthesis: Motivation

- For control-logic applications, 3 levels seem to be a good trade-off between the speed of two-level implementation and the density of multi-level one
  - speed in crucial for control logic:
    - IBM's Gigahertz processor has been implemented on a single PLA (1998)
- Algorithms for 3-level optimization are much simpler than for multi-level

Formulation of the problem

input: a Boolean function $f(x_1,x_2, \ldots,x_n)$

output: an expression for f of type

$$f(x_1,\ldots,x_n) = (P_1+ \ldots+P_k)\bullet(P_{k+1}+ \ldots +P_r)$$

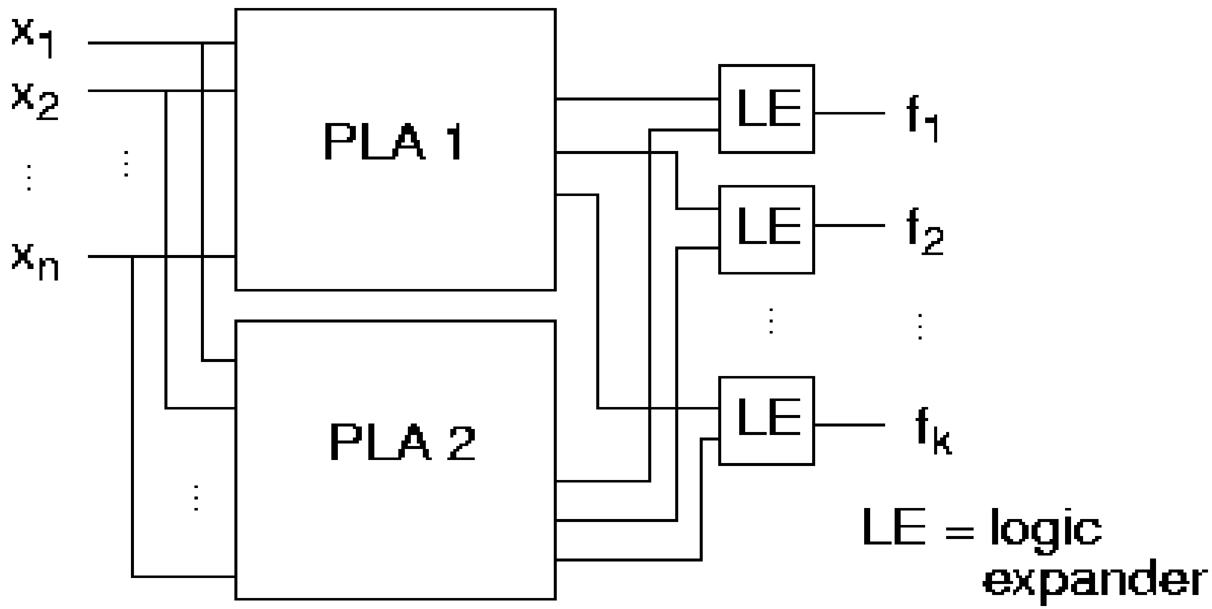
with the minimal number r of products P_i, where "•" is a suitable binary operation

Previous work

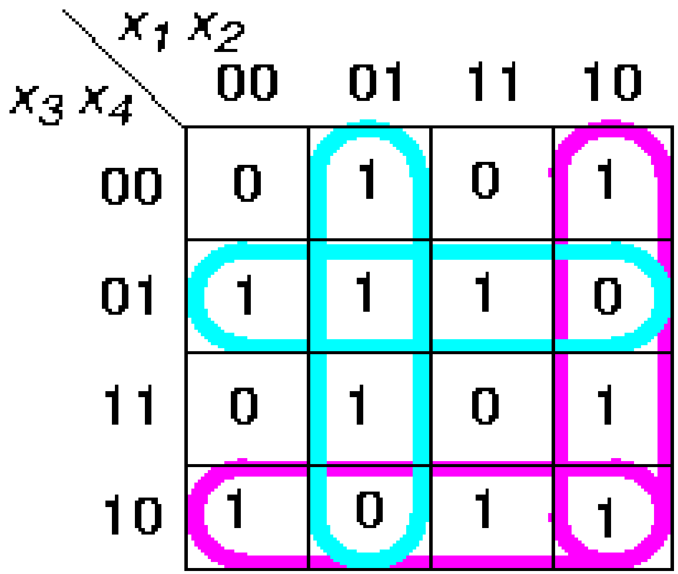
- Fast heurictics for finding a close to minimal expression for a specified "•"
  - 1991: AND/OR
  - 1995: XOR
- Fast heuristic for finding a close to minimal expression for any "•" (2000):

Implementation

- The expression of the above type can be implemented by the following Programmable Logic Device:



Example: • = XOR



$P_1 = \overline{x}_1 x_2$

$P_2 = \overline{x}_3 x_4$

$P_3 = x_1 \overline{x}_2$

$P_4 = x_3 \overline{x}_4$

$$f(x_1, x_2, \ldots, x_n) = (P_1 + P_2) \oplus (P_3 + P_4)$$

Example: • = AND

$$f = g_1 \cdot g_2$$

	00	01	11	10	10	11	01	00
00	1				1			
01		1				1		
11			1				1	
10				1				1

g_1

	00	01	11	10	10	11	01	00
00	1	1			1	1		
01	1	1			1	1		
11			1	1			1	1
10			1	1			1	1

g_2

	00	01	11	10	10	11	01	00
00	1			1	1			1
01		1	1			1	1	
11		1	1			1	1	
10	1			1	1			1

Example: • = OR

$$f(x_1,...,x_n) = g_1(x_1,...,x_{n/2}) + g_2(x_{n/2+1},...,x_n)$$

– same number of products as in the minimal sum-of-products expression, but they are distributed in a way favorable for the total area of the targeted PLD

– the goal is to divide the products so that the number of common inputs and outputs in g_1 and g_2 is minimal

AND-OR-XOR Algorithm

- We want to construct g and h such that:

$$f = g \oplus h$$

- Conditions which should be satisfied for this:
  - Each cube of F_f should belong to either F_g or F_{h}, but not both
  - Some of the cubes of R_f may belong to both F_g and F_h

h \ g	0	1
0	0	1
1	1	0

Main steps of the algorithm

- Cluster the cubes of F_f into equivalence classes
  - If two cubes intersect, they belong to the same class
- Partition the resulting clusters into two groups
  - These groups are the initial on-sets of g and h
- Find cover for the incompletely specified function with the on-set F_g, don't care set $R_f \cup D_f$, and off-set F_h

Example

x_3x_4 \ x_1x_2	00	01	11	10
00	0	1	0	1
01	1	1	1	0
11	0	1	0	1
10	1	0	1	1

g

x_3x_4 \ x_1x_2	00	01	11	10
00	-	1	-	0
01	1	1	1	-
11	-	1	-	0
10	0	-	0	0

h

x_3x_4 \ x_1x_2	00	01	11	10
00		0		1
01	0	0	0	
11		0		1
10	1		1	1

Main steps of the algorithm, cont.

- Compute the intersection of the resulting cover with R_f. Add it to the on-set of h
- Find cover for the incompletely specified function with the on-set F_h, don't care set $R_f \cup D_f$, and off-set F_{g_init}

h

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00	-	0	-	1
01	0	0	0	1
11	-	0	-	1
10	1	1	1	1

AND-OR-AND Algorithm

- We want to construct g and h such that:

$$f = g \cdot h$$

- Conditions which should by satisfied for this:
  - $F_g \cap F_h = F_f$
  - $R_g \cup R_h = R_f$

h \ g	0	1
0	0	0
1	0	1

- We can use some of the cubes in the off-set of g or h (but not both) to reduce the size of the cover of F_f

Example

x_3x_4 \ x_1x_2	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

g

1	1	0	0
1	1	0	0
0	0	1	1
0	0	1	1

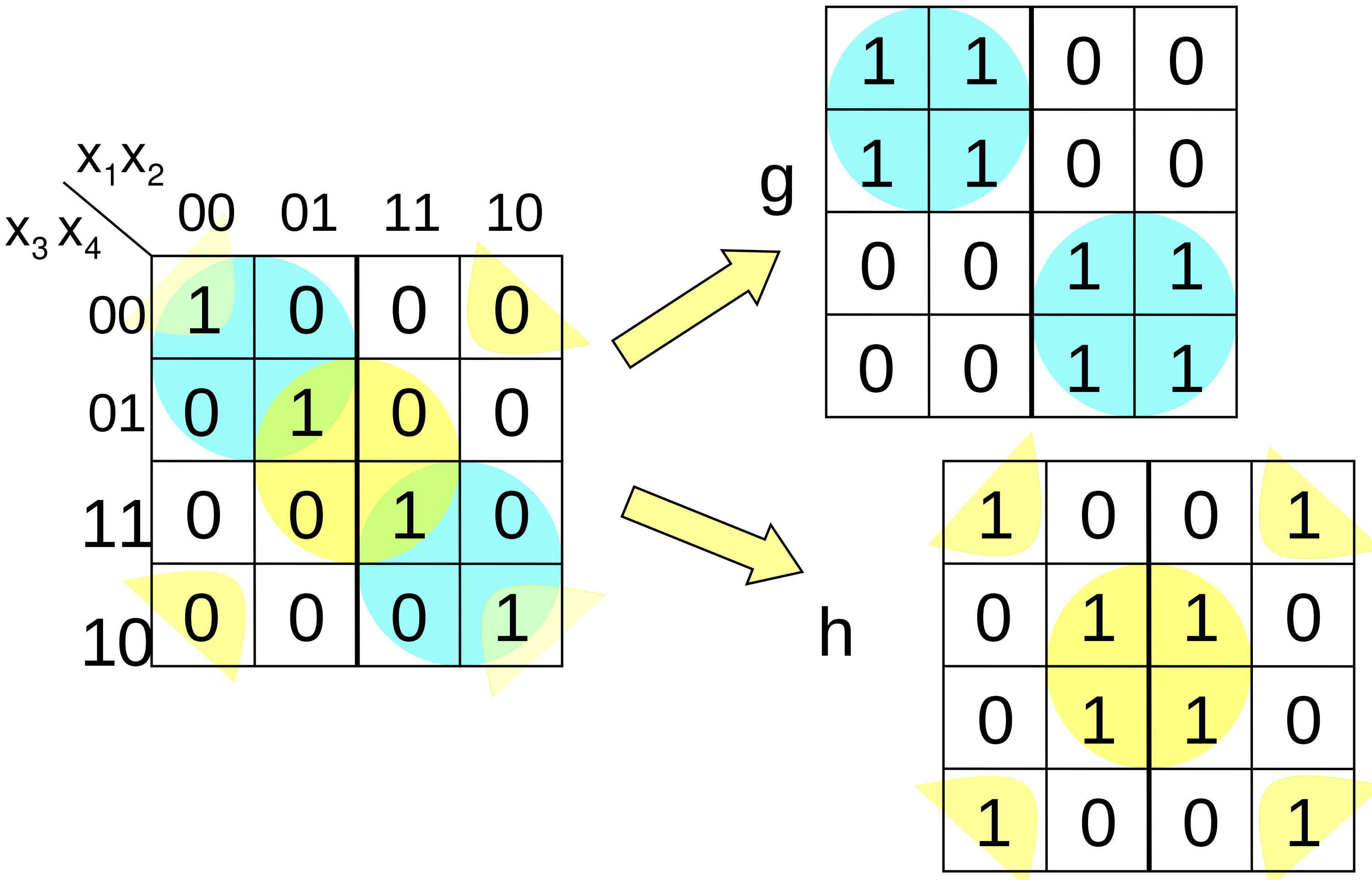
h

1	0	0	1
0	1	1	0
0	1	1	0
1	0	0	1

Main steps of the algorithm

- Select a pair of cubes $c_1, c_2 \in F_f$ and compute their supercube, **$sup(c_1,c_2)$.** Add $sup(c_1,c_2)$ to F_g
- Mark cubes of the off-set R_f which are contained in $sup(c_1,c_2)$ in blue color
- Select a cube $c_3 \in F_f$ such that **$sup(c_2,c_3)$** does not contain any cubes of R_f marked in blue. Add $sup(c_1,c_2)$ to F_h
- Mark cubes of the R_f which are contained in $sup(c_2,c_3)$ in yellow color
- Repeat the above steps until each cube of F_f is included in two supercubes with different colors

Example


x_1x_2
x_3x_4
00 01 11 10
00 1 0 0 0
01 0 1 0 0
11 0 0 1 0
10 0 0 0 1
g
1 1 0 0
1 1 0 0
0 0 1 1
0 0 1 1
h
1 0 0 1
0 1 1 0
0 1 1 0
1 0 0 1

Summary

- Three-level decomposition algorithms can be repeatedly applied to the get a multi-level circuit for a given functions
- They are more complex then two-level minimization algorithms, but more simple than multi-level optimization algorithms