

California Institute of Technology
Department of Computer Science
Electronic Design Automation

CS137a, Winter 2004

Assignment #2

Wednesday, January 21

Due: Friday, February 6, 11:59pm.

Resources You are free to use any books, articles, notes, or papers as references. Provide citations in your writeup as appropriate.

Collaboration Please work independently on this assignment.

Writeup Writeup should be in an electronically readable format (HTML or PDF preferred). State any assumptions you need to make.

Tools You should work 1–4 without using tools and show your work. You are welcome to *check* your work using tools. See notes on 5 about tool usage on its various parts.

- You can find `espresso` and `sis` in `/cs/research/ic/software/bin/`

Problems

1. What is the minimum 2-level logic implementation of:

$$f = \bar{a} \cdot b \cdot \bar{c} \cdot \bar{d} + \bar{a} \cdot b \cdot c \cdot \bar{d} + \bar{a} \cdot b \cdot c \cdot d + a \cdot \bar{b} \cdot c \cdot \bar{d} + a \cdot b \cdot \bar{c} \cdot \bar{d} + a \cdot b \cdot c \cdot \bar{d} + a \cdot b \cdot c \cdot d$$

- (a) generate all primes
- (b) show the initial prime implicant table
- (c) show final, optimal cover

2. Find the optimal Cover for the following Prime Implicant Table.

- Identify Essential Primes in this first table
- Identify all Dominating Rows in this first table
- Identify all Dominating Columns in this first table
- Show the Reduced Table and identify any cover selections made at this stage
- Continue Reducing, Branching as necessary
- Report an optimal cover

	A	B	C	D	E	F	G	H	I
a		X	X			X	X		
b			X	X					
c					X				
d					X	X			
e							X	X	X
f								X	X
g	X							X	
h	X								X

3. [Problem 5, page 217 in [4]] Generate all the kernels of

$$f = a \cdot c \cdot e + a \cdot c \cdot g + b \cdot c \cdot e + b \cdot c \cdot g + a \cdot d \cdot e + a \cdot d \cdot g + b \cdot d \cdot e + b \cdot d \cdot g$$

by using the kernel generation algorithm of Figure 7.1 and alternately by constructing the cube-literal matrix and generating all the prime rectangles using the procedure in Figure 7.3.

4. [Problem 8, page 217 in [4]] Perform Boolean division and compute $f//g$ given the logic functions f and g below.

$$f = a \cdot b + \bar{a} \cdot \bar{c} \cdot d + a \cdot c \cdot d + \bar{a} \cdot \bar{b} \cdot c \cdot \bar{d}$$

$$g = c \cdot \bar{d} + \bar{c} \cdot d$$

5. Provide two state assignments for the following.
- One should minimize the number of product terms in the 2-level (PLA) implementation. [You should be able to do this based on the lecture/reading without using any tools; you may use tools to check your answer.]
 - The second should use no more than 3 state bits and should be chosen to maximize the product terms. (This asks you to see how **bad** of an assignment you can come up with.) [Feel free to use tools however you'd like for this one.]
 - Identify the number of product terms required by each after two-level optimization. [You are welcome to use **espresso** to do two-level optimization for this.]

```

0 START state6 00
0 state2 state5 00
0 state3 state5 00
0 state4 state6 00
0 state5 START 10
0 state6 START 01
0 state7 state5 00
1 state6 state2 01
1 state5 state2 10
1 state4 state6 10
1 state7 state6 10
1 START state4 00
1 state2 state3 00
1 state3 state7 00

```

6. Provide an algorithm for two-level PLA mapping that will guarantee to cover every product term at least twice. Simply making two copies of the solution you would get from the normal, ESPRESSO-EXACT algorithm would cover every product term twice. However, many of the minterms are already covered twice as a result of the base algorithm. Therefore, in general, it should not be necessary to duplicate all of the product terms. Your algorithm should try to minimize the number of product terms.
- Develop the algorithm. (Give this 1–2 hours of thinking *after* you fully understand the ESPRESSO-EXACT algorithm. Writeup the best algorithm you come up with in that time frame.)
 - Give pseudocode for your algorithm in the style of [1] or [2].
 - What is the worst-case running time of your algorithm?
 - Is your solution optimal? Sketch why or why not.

[**Motivations:** If the most likely fault in a molecular-switch PLA is that a connected switchpoint becomes disconnect, this would guarantee that every minterm was covered by two connections in the product array. A generalization of this (n -cover minterms, where n depends on the number of outputs using the product term) could be used for fanout bounding which will be important for achieving high speed operation for nanoPLAs [3].]

References

- [1] Jason Cong and Yuzheng Ding. On area/depth trade-off in lut-based fpga technology mapping. *IEEE Transactions on VLSI Design*, 2(2):137–148, June 1994.
- [2] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [3] André DeHon and Michael J. Wilson. Nanowire-Based Sublithographic Programmable Logic Arrays. February 2004.
- [4] Srinivas Devadas, Abhijit Ghosh, and Kurt Keutzer. *Logic Synthesis*. McGraw-Hill, New York, 1994.