CSEE 6861 CAD of Digital Systems
Handout: Lecture #1
1/21/16

Prof. Steven M. Nowick
nowick@cs.columbia.edu

Department of Computer Science (and Elect. Eng.)
Columbia University
New York, NY, USA

Overview of Design Flow

Key Synthesis/Optimization Steps: at 3 Levels

1. Architectural Synthesis (also, "High-Level Synthesis" [HLS])

Starting point: behavioral system specification

Steps: ***scheduling, resource allocation (sharing) and binding***

Outcome: register-transfer level (RTL)] optimized design
for block-level datapath + FSM controller specification

2. Logic Synthesis

Steps:

sequential synthesis: *FSM optimization*

combinational synthesis: *(i) 2-level logic minimization, (ii) multi-level logic optimization*

technology mapping: *optimal mapping of gates to VLSI "library" cells*

Outcome: mapped gate-level circuit

3. Physical Design

Steps: ***circuit partitioning, chip floorplanning, place-and-route ("P&R")***
... *+ late timing correction/optimizations, etc.*

Outcome: complete chip layout ➔ ready for fabrication

#3

Architectural Synthesis

High-Level Specification: Differential Equation Solver *(diff-eq)* Custom Unit

```
diffeq {
        read ( x, y, u, dx, a );
        repeat  {
                xl  =   x + dx;
                ul  =   u - ( 3 * x * u * dx ) - ( 3 * y * dx );
                yl  =   y + u * dx;
                c   =   xl <  a;
                x = xl ; u = ul ; y = yl;
                }
        until ( c );
        write ( y );
}
```

Figures courtesy of: G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill (1994)

#4

Architectural Synthesis

Target Micro-Architecture: Register-Transfer Level

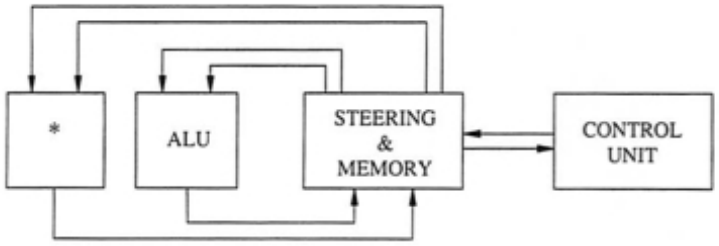


FIGURE 1.12
Example of structural view at the architectural level.

#5

Architectural Synthesis

Detailed Target Micro-Architecture:

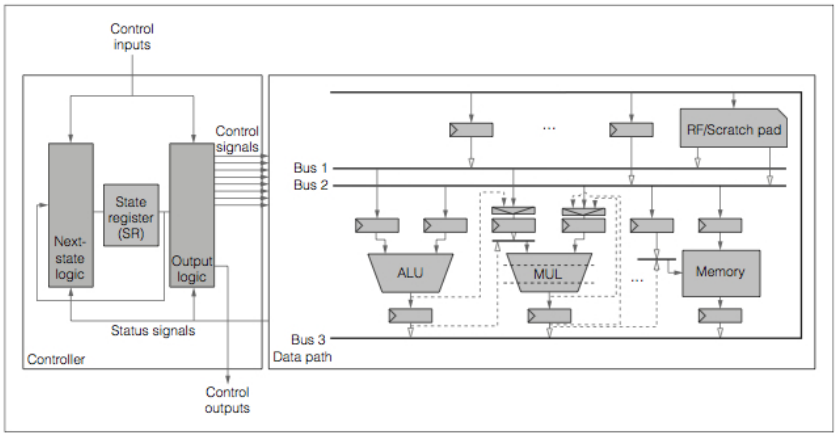


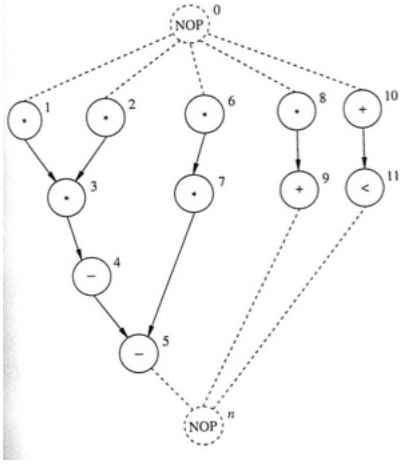
Figure 2. Typical architecture.

Courtesy of: P. Coussy, D.D. Gajski, M. Meredith and A. Takach,
"An Introduction to High-Level Synthesis", IEEE Design & Test of Computers (July/Aug. 2009)

#6

Architectural Synthesis

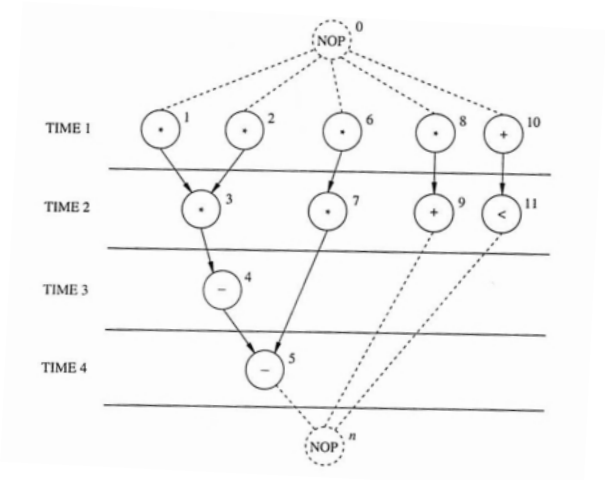
Unscheduled Control-Dataflow Graph (CDFG): *diff-eq*



#7

Architectural Synthesis

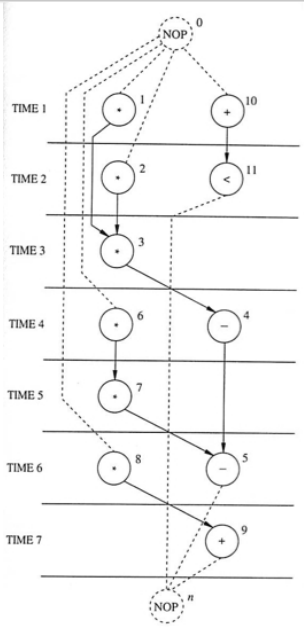
Scheduled CDFG: minimum-latency



#8

Architectural Synthesis

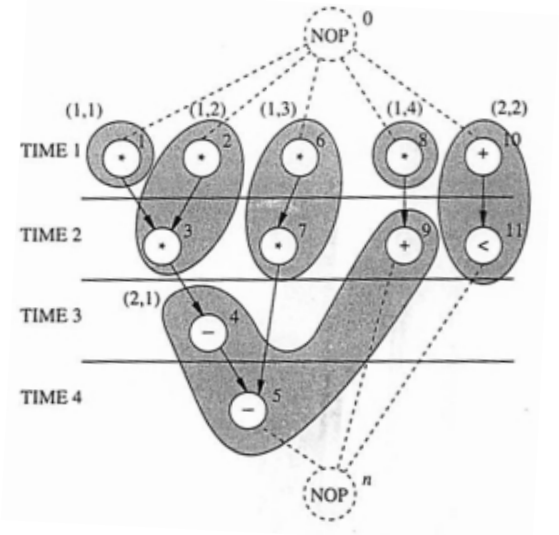
Scheduled CDFG: min-area
= resource-constrained (RC)



#9

Architectural Synthesis

Resource Allocation/Sharing



#10

Logic Synthesis

Initial Unoptimized Combinational Function Block: *"opa"*

17 inputs/69 outputs ➔ 2119 gate inputs (literals)

```
.i 17
.o 69
011001----------
10000000000000000000000000000000000000000000000000000000000000000000000
011100----------
10000000000000000000000000000000000000000000000000000000000000000000000
01110-1---------
10000000000000000000000000000000000000000000000000000000000000000000000
0100111---------
10000000000000000000000000000000000000000000000000000000000000000000000
01-1001---------
10000000000000000000000000000000000000000000000000000000000000000000000
0011-11------0---
10000000000000000000000000000000000000000000000000000000000000000000000
00-111----------
10000000000000000000000000000000000000000000000000000000000000000000000
00-1000---------
10000000000000000000000000000000000000000000000000000000000000000000000
0--1110---------
10000000000000000000000000000000000000000000000000000000000000000000000
011-0-----------
01000000000000000000000000000000000000000000000000000000000000000000000
011---0---------
01000000000000000000000000000000000000000000000000000000000000000000000
01---0----------
01000000000000000000000000000000000000000000000000000000000000000000000
000--1----------
01000000000000000000000000000000000000000000000000000000000000000000000
000---1---------
01000000000000000000000000000000000000000000000000000000000000000000000
00-1------------
01000000000000000000000000000000000000000000000000000000000000000000000
00--1-----------
01000000000000000000000000000000000000000000000000000000000000000000000
0-1--0----------
01000000000000000000000000000000000000000000000000000000000000000000000
0-11---10---0-000
01000000000000000000000000000000000000000000000000000000000000000000000
0-11----01--0-000
01000000000000000000000000000000000000000000000000000000000000000000000
0-001-----------
01000000000000000000000000000000000000000000000000000000000000000000000
0-00--1---------
01000000000000000000000000000000000000000000000000000000000000000000000
0--10-----------
01000000000000000000000000000000000000000000000000000000000000000000000
```

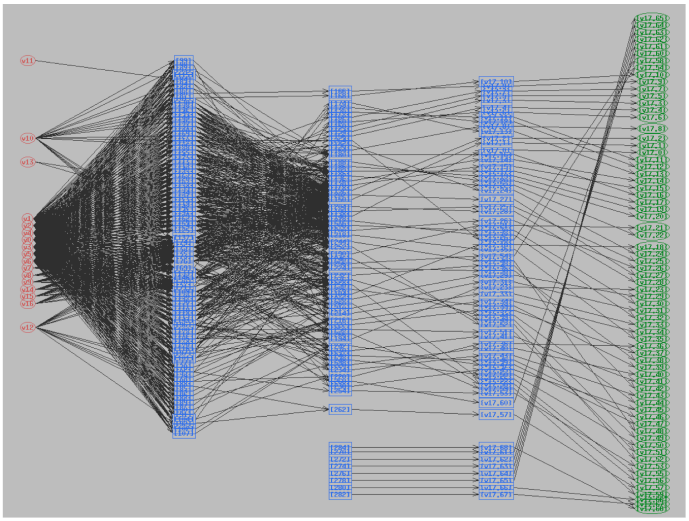
Small portion of input specification

#11

Logic Synthesis

Result of Heuristic 2-Level Minimization: *"opa"*

2119 gate inputs (literals) down to 1153 gate inputs

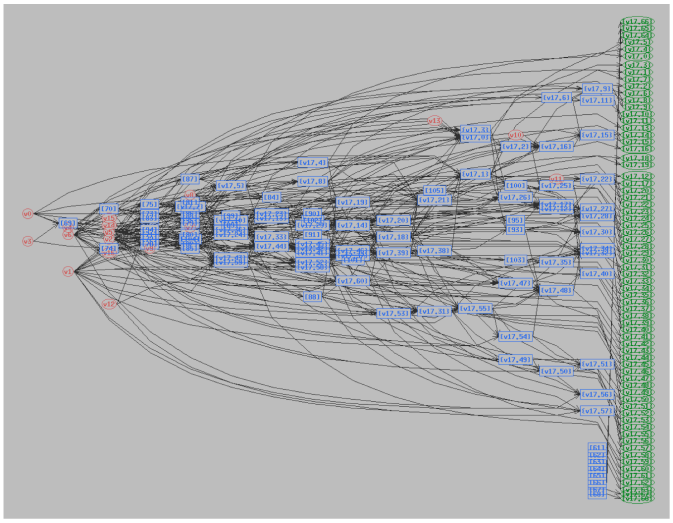


#12

Logic Synthesis

Result of Multi-Level Optimization: *"opa"*

2119 gate inputs (literals) down to 430 gate inputs



#13

Logic Synthesis

Technology Mapping = binding to VLSI cells

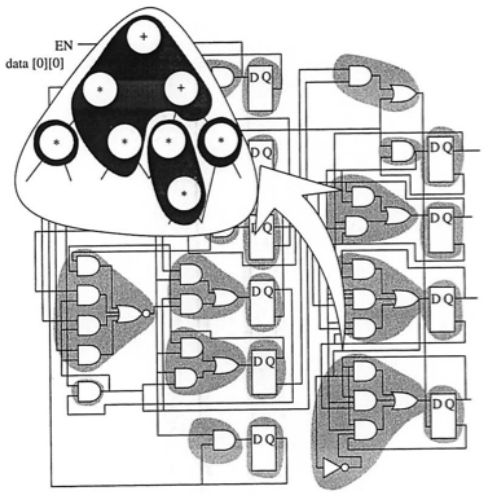


FIGURE 10.6
Covering of a subject graph.

#14

Physical Design

Optimal Circuit Partitioning: Kernighan-Lin Algorithm

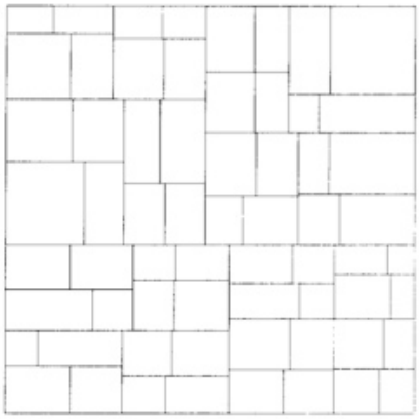


Fig. 1. Partitioning of 412-cell circuit into groups of size ⩽8.

Figure courtesy of: A.E. Dunlop and B.W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits", IEEE Trans. On Computer-Aided Design (Jan. 1985)

#15

Physical Design

Optimal Place-and-Route

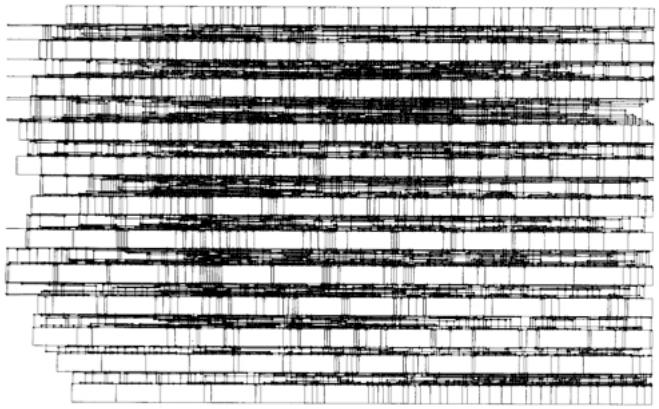


Fig. 4. Automatically generated layout for circuit with 453 signals and 412 cells.

Figure courtesy of: A.E. Dunlop and B.W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits", IEEE Trans. On Computer-Aided Design (Jan. 1985)

#16

Physical Design

Final Chip Layout

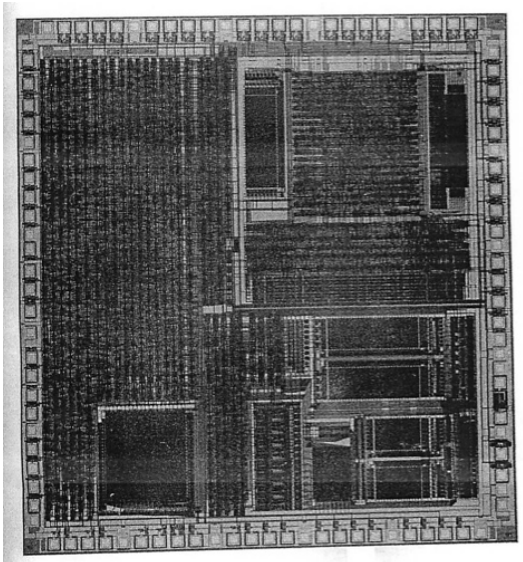


FIGURE 1.3
One of AT&T's Application Specific Standard Product chips. The chip was designed and laid out using AT&T CAD tools, with a standard cell design style. (Courtesy of AT&T.)

17

Review: Basic Definitions (2-Level Logic Minimization)

Review: Basic Definitions

Literal: a variable (x) or its complement (x')

Product: an "AND" of literals (e.g. xy' z, a' bcd')

Cube: a product (another equivalent name)

Implicant: a cube/product which contains no OFF-set minterm (i.e. 0 value)

- *Note:* implicants do not need to contain any ON-set minterms (i.e. 1 values), but they usually do

Prime Implicant (PI, prime): a maximal implicant (i.e. not contained in any larger implicant)

Essential Prime Implicant (essential): a prime which contains at least one ON-set minterm (i.e. 1 value) not contained in any other prime

Sum-of-products (SOP, disjunctive normal form): a sum of products ("AND-OR" 2-level circuit)

Cover: a set of primes (SOP) which together contain all ON-set minterms (i.e. 1 values) of a function

Complete Sum: a cover containing all possible prime implicants of the function

#19

Review: Basic Definitions

The 2-Level Logic Minimization Problem: given Boolean function f,

(i) Find a minimum-cost set of prime implicants which "covers" (i.e. contains) all ON-set minterms of function f (and possibly some DC-set minterms)

or *(…equivalently):*

(ii) Find a minimum-cost cover F of function f

#20

2-Level Logic Minimization: Definitions + Design Space Exploration

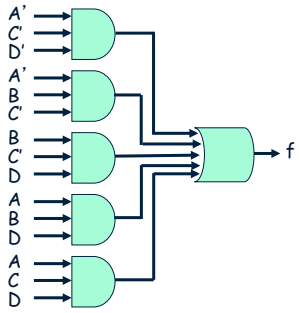
2-Level Logic Minimization: Example

CD \ AB	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	0	0	0	0

2-Level Logic Minimization: Example

Solution #1: All Primes = 5 Products (AND gates)

CD \ AB	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	0	0	0	0



"Complete Sum":
cover containing all prime implicants

corresponding 2-level implementation

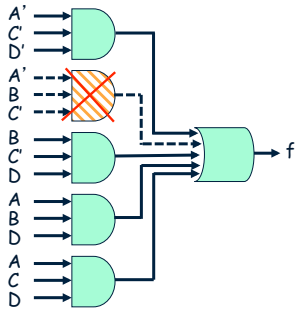
#23

2-Level Logic Minimization: Example

Solution #2: Subset of Primes = 4 Products (AND gates)

CD \ AB	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	0	0	0	0

Locally sub-optimal solution



"Redundant Cover":
can remove a product and still have legal cover

corresponding 2-level implementation

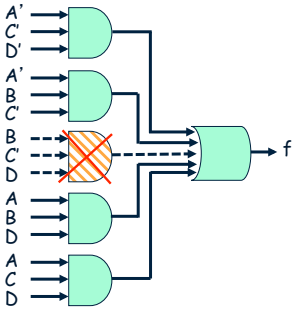
#24

2-Level Logic Minimization: Example

Solution #3: Subset of Primes = 4 Products (AND gates)

CD \ AB	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	0	0	0	0

Locally optimal solution:
… but globally sub-optimal = "LOCAL MINIMUM"



"Irredundant Cover" (but globally sub-optimal):
cannot remove any product and still have legal cover

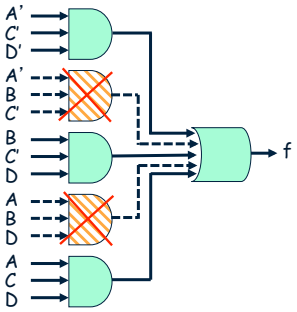
corresponding 2-level implementation

2-Level Logic Minimization: Example

Solution #4: Subset of Primes = 3 Products (AND gates)

CD \ AB	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	0	0	1	1
10	0	0	0	0

Globally-optimal solution



corresponding 2-level implementation

OPTIMAL SOLUTION (also irredundant)

Exact 2-Level Logic Minimization: Quine-McCluskey (QM) Method

Quine-McCluskey Method: Examples

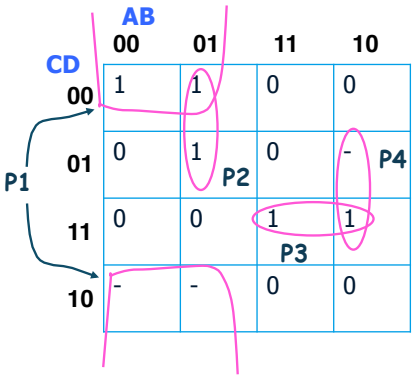
Example #1: f(A,B,C,D) = m(0,4,5,11,15) + d(2,6,9)

[m = ON-set minterms, d = DC-set minterms]

CD \ AB	00	01	11	10
00	1	1	0	0
01	0	1	0	-
11	0	0	1	1
10	-	-	0	0

Quine-McCluskey Method: Examples

Example #1 (cont.)

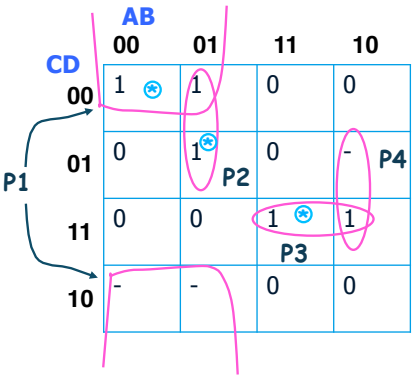


Generate all prime implicants

#29

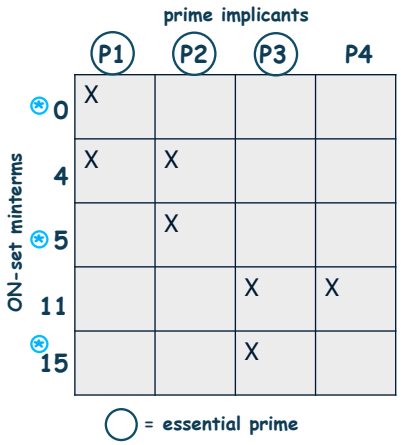
Quine-McCluskey Method: Examples

Example #1 (cont.)



⊛ = distinguished minterm

Prime Implicant Table



◯ = essential prime

Approach: remove & save essentials {p1, p2, p3}, and delete intersecting rows … *empty table: nothing left to cover*.

#30

Quine-McCluskey Method: Examples

Example #2: $f(A,B,C) = m(0,1,2,6) + d(5)$

[m = ON-set minterms, d = DC-set minterms]

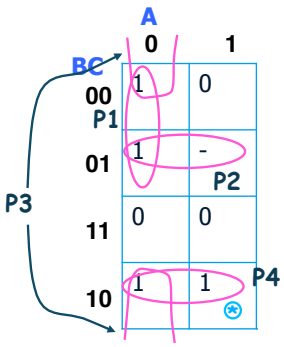
BC \ A	0	1
00	1	0
01	1	-
11	0	0
10	1	1

More complex example: illustrates "table reduction step" using column dominance

Quine-McCluskey Method: Examples

Example #2: $f(A,B,C) = m(0,1,2,6) + d(5)$

[m = ON-set minterms, d = DC-set minterms]



⊛ = distinguished minterm

Prime Implicant Table

ON-set minterms \ prime implicants	P1	P2	P3	P4
0	X		X	
1	X			X
2		X	X	
⊛ 6		X		

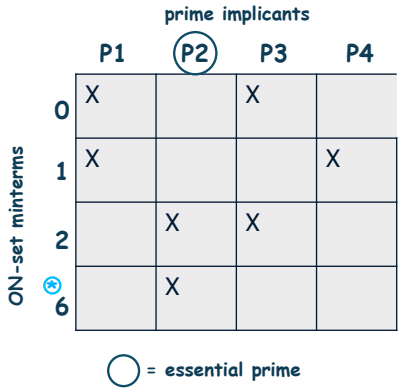
◯ = essential prime

Initial PI Table

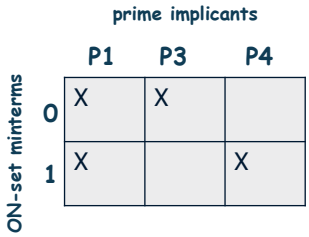
Quine-McCluskey Method: Examples

Example #2: $f(A,B,C) = m(0,1,2,6) + d(5)$

[m = ON-set minterms, d = DC-set minterms]



Initial PI Table



Reduced PI Table (a)

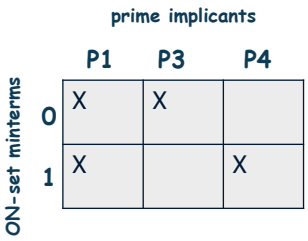
Approach: remove & save essential p2, and delete intersecting rows.

#33

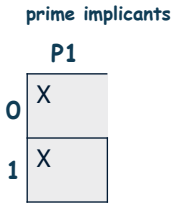
Quine-McCluskey Method: Examples

Example #2: $f(A,B,C) = m(0,1,2,6) + d(5)$

[m = ON-set minterms, d = DC-set minterms]



Reduced PI Table (a)



Reduced PI Table (b)

"Column Dominance":
- column p1 'column-dominates' column p3
- column p1 'column-dominates' column p4

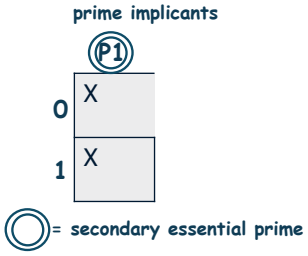
…*delete dominated columns {p3,p4}*

#34

Quine-McCluskey Method: Examples

Example #2: $f(A,B,C) = m(0,1,2,6) + d(5)$

[m = ON-set minterms, d = DC-set minterms]



"Secondary Essential Primes":
- column p1 *has now become 'essential'*

Approach: remove & save secondary essential p1, and delete intersecting rows.
… *empty table: nothing left to cover*.

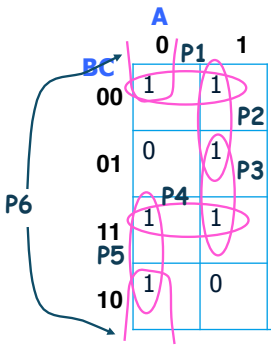
Reduced PI Table (b)

Final solution: {p1, p2}

Quine-McCluskey Method: Examples

Example #3: $f(A,B,C) = m(0,2,3,4,5,7)$

[m = ON-set minterms, d = DC-set minterms]



More complex example: illustrates (i) no reduction possible, and (ii) resulting "cyclic core"

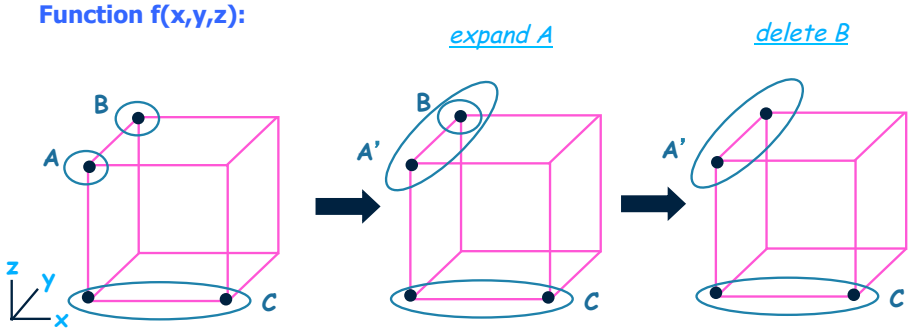
FOR EXACT SOLUTION: can use Petrick's Method (or more advanced techniques)
SEE QUINE-MCCLUSKEY HANDOUT

Heuristic 2-Level Logic Minimization: the "Espresso" Method

Introduction to ESPRESSO: Examples

Example #1: Basic "Expand" Step

Function f(x,y,z):



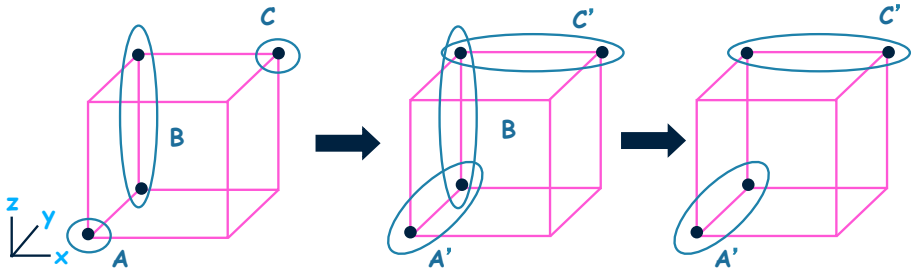
Initial cover ("seed")

#38

Introduction to ESPRESSO: Examples

Example #2: Basic "Expand" + "Irredundant" Steps

Function f(x,y,z):



Initial cover ("seed")

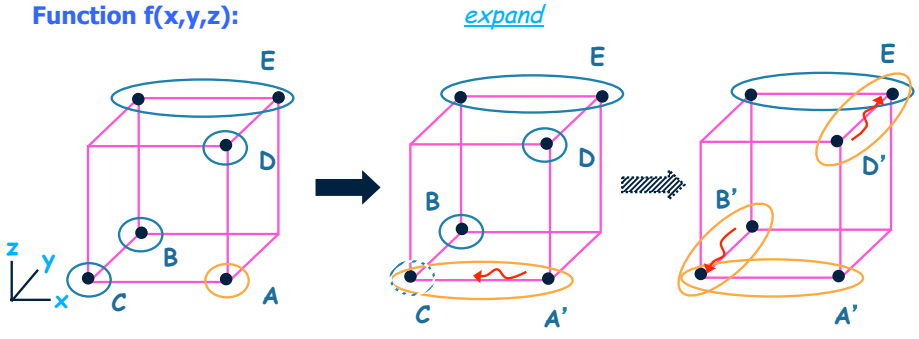
"Expand" Step

"IRRED" Step

Introduction to ESPRESSO: Examples

Example #3: "Expand"/"Irredundant"/"Reduce" Iteration

Function f(x,y,z):



Initial cover ("seed")

expansion order: A, ... D, B

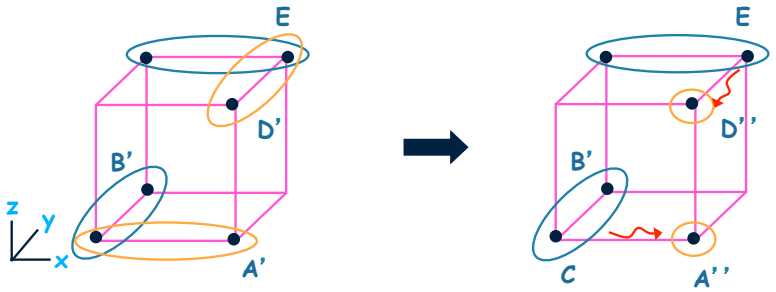
C fully contained in A': delete C

Introduction to ESPRESSO: Examples

Example #3 (cont.): "Expand"/"Irredundant"/"Reduce" Iteration

"irredundant": no change

reduce



Result of "expand" step (copied)

cube order: A', D'

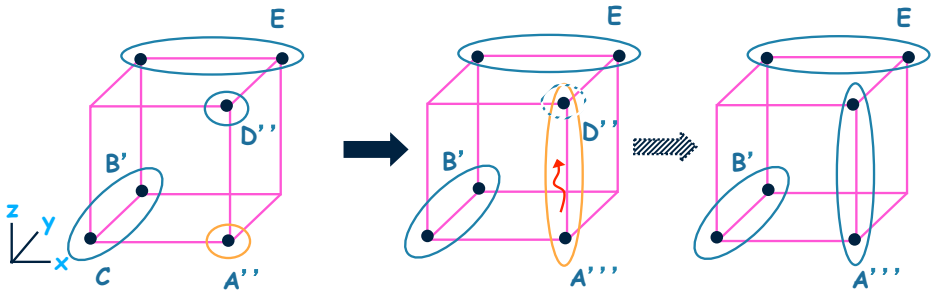
#41

Introduction to ESPRESSO: Examples

Example #3 (cont.): "Expand"/"Irredundant"/"Reduce" Iteration

expand (2nd time!)

final cover



Result of "reduce" step (copied)

expand order: A'', (D'')

D'' fully contained in A''': delete D''

#42