0	Admin
1	Overview
2	Simplification
3	State assignment

1 OVERVIEW

a) Optimization/minimization – "Style"

Robust design more important

50% more gates, but works 100%

More important than fewer gates, but 99% reliable

Real world – VHDL algorithm design, not gate interconnect

b) Robustness details:

metastability

testability

error correction

2. SIMPLIFICATION

Why?

- Reduce HW complexity
- Save gates/space
- Fewer routing resources
- Improve speed, etc.

How?

- Crank and grind
- Adjacency theorem – $X \cdot Y + X \cdot \overline{Y} = ? X$
- Graphical form of adjacency theorem

2.1 Crank and Grind

Example: $X = A + ABC + \overline{A}C = A(1+BC) + \overline{A}C$

$= A + \overline{A}C = A + C$

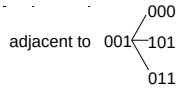
if A not true, then $\overline{A}C = 1 \cdot C = C$

2 SIMPLIFICATION (cont.)

2.2 Adjacency Theorem

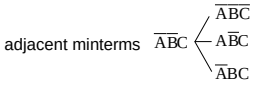
Adjacency: "Distance" between binary strings = 1

distance (001,000) = 1



Note: only 1 bit changes

"Distance" is important for error detection and correction



2 SIMPLIFICATION (cont.)

2.2 Adjacency theorem (cont.)

$\text{distance} \triangleq \sum_{i=0}^{N-1} a_0 \oplus b_i$ binary vector $\underline{a}$

$a_{N-1}, \ldots, a_0 \quad a_i \in \{0,1\}$

Example using adjacency theorem only

$X = A + ABC + \overline{A}C = A[\overline{B}\overline{C} + \overline{B}C + B\overline{C} + BC] + ABC + \overline{A}C[B + \overline{B}]$

expand and factor

$C[AB + A\overline{B} + \overline{A}B + \overline{A}\overline{B}]$

$+ A[\overline{B}\overline{C} + \overline{B}C + B\overline{C} + BC + BC]$

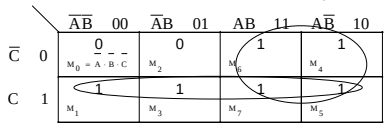
$= C \cdot [1] + A \cdot [1] = A + C$ ✓

2 SIMPLIFICATION (cont.)

2.3 Karnaugh Map

$X = \sum(1,3,4,5,6,7)$ minterms

	ABC	X
M_0	000	0
	001	1
	010	0
	011	1
	100	1
	101	1
	110	1
M_7	111	1



Note: 01 and 10 are not adjacent

$X = \sum(1,3,7,5) = C[\overline{A}\overline{B} + \overline{A}B + AB + A\overline{B}]$

$+ \sum(4,5,6,7) + A[\overline{B}\overline{C} + \overline{B}C + B\overline{C} + BC]$

$= A + C$ ✓

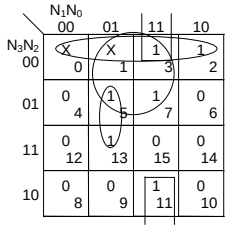
 Copyright © 1999 The Regents of the University of California

2 SIMPLIFICATION (cont.)

2.4 Four-variable K map example

Design minimized logic for a function PRIME.H, which is asserted if a 4-bit number *N* is prime, except N=0 and N=1 are don't care

$\text{PRIME} = \sum(2,3,5,7,11,13)$



$\text{PRIME} = \overline{N}_3\overline{N}_2 + \overline{N}_3 \cdot N_0 + N_2 \cdot \overline{N}_1 \cdot N_0 + \overline{N}_2 \cdot N_1 \cdot N \cdot 0$

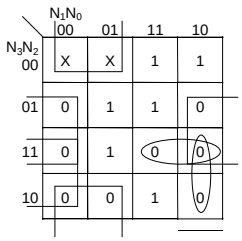
$\sum(1,3,5,7) = \overline{N}_3 N_0(\overline{N}_2\overline{N}_1 + N_2\overline{N}_1 + \overline{N}_2 N_1 + N_2 N_1) = \overline{N}_3 N_0$

 Copyright © 1999 The Regents of the University of California

2 SIMPLIFICATION (cont.)

2.5 P.O.S. form

$\overline{\text{PRIME}} = \sum(4,6,8,9,10,12,14,15) + D(0,1)$ circle "0"s



$\overline{\text{PRIME}} = \overline{N}_2 \cdot \overline{N}_1 + N_2\overline{N}_0 + N_3 N_2 \cdot N_1 + N_3\overline{N}_0$

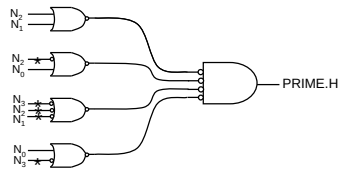
by DeMorgan

$\text{PRIME} = \overline{\overline{\text{PRIME}}} = (N_2 + N_1)(\overline{N}_2 + N_0)(\overline{N}_3 + \overline{N}_2 + \overline{N}_1)(\overline{N}_3 + N_0)$

 Copyright © 1999 The Regents of the University of California

2 SIMPLIFICATION (cont.)

2.5 P.O.S. form (cont.)



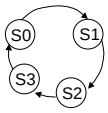
	N_3	N_2	N_1	N_0					
Check 3	0	0	1	1	(1)	(1)	(1)	(1)	=1 ✓
Check 8	1	0	0	0	(0)	(1)	(1)	(0)	=0 ✓

 Copyright © 1999 The Regents of the University of California

3 FINITE STATE MACHINE STATE ASSIGNMENT

Goal: Choose state assignment to simplify logic design for NSD/OD

3.1 Example: "One Hot" encoding



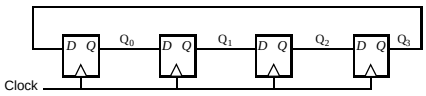
not minimal number of FF

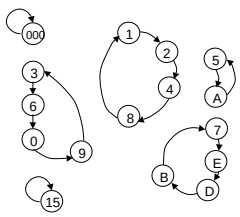
P_S Q_3 Q_0	N_S	
(S_0) 0001	(S_1) 0010	$D_0 = Q_3$
(S_1) 0010	(S_2) 0100	$D_1 = Q_0$
(S_2) 0100	(S_3) 1000	$D_2 = Q_1$
(S_3) 1000	(S_4) 0001	$D_3 = Q_2$

 Copyright © 1999 The Regents of the University of California

3 FINITE STATE MACHINE STATE ASSIGNMENT (cont.)

Simple logic



How many states? 16. What happens to other states? Possible trouble if upset.

PS	NS
0000	0000
0001	0010
0010	0100
0011	0110
0100	1000
0101	1010
0110	1100
0111	1110
1000	0001
1001	0011
1010	0101
1011	0111
1100	1001
1101	1011
1110	1101
1111	1111