

## Topic #4 – combinational circuit design

### Task:

Given a description of problem (logical statement), find the corresponding digital circuits that produce the output (answer) given a set of inputs (condition).

### Examples:

- Parking lot controller (Lab 4)
- Elevator controller
- Prime number indicator
- Adder, subtractor, ...

S. Jay Yang, CE, RIT

## Design example: alarm controller

### Problem statement:

- The ALARM output is 1 if PANIC is 1, or if ENABLE is 1 and the house is not secure.
- The house is secure if WINDOW, DOOR, GARAGE are all 1

### This can be put in logic expressions as follows:

$ALARM = PANIC + ENABLE \cdot SECURE'$   
 $SECURE = WINDOW \cdot DOOR \cdot GARAGE$   
 $ALARM = PANIC + ENABLE \cdot (WINDOW \cdot DOOR \cdot GARAGE)'$

Multiply out and use (T13), we get the SoP form

$$ALARM = PANIC + ENABLE \cdot WINDOW' + ENABLE \cdot DOOR' + ENABLE \cdot GARAGE'$$

S. Jay Yang, CE, RIT

## Brute-force approach

- Design: given a description or truth table, find the corresponding Boolean expression and digital circuit.

### Brute-force design methodology:

- Truth table → canonical sum
- → SoP or sum of minterms
- → AND-OR / NAND-NAND

### Example: prime number detector

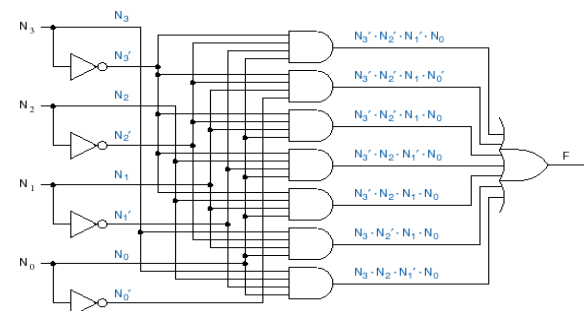
$$F = \sum_{N_3 N_2 N_1 N_0} (1, 2, 3, 5, 7, 11, 13)$$

Row	$N_3$	$N_2$	$N_1$	$N_0$	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

S. Jay Yang, CE, RIT

## Minterm list -> canonical sum

$$\begin{aligned}
 F &= \sum_{N_3 N_2 N_1 N_0} (1, 2, 3, 5, 7, 11, 13) \\
 &= N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0' + N_3' \cdot N_2' \cdot N_1 \cdot N_0 + N_3' \cdot N_2 \cdot N_1' \cdot N_0 \\
 &\quad + N_3' \cdot N_2 \cdot N_1 \cdot N_0 + N_3 \cdot N_2' \cdot N_1 \cdot N_0 + N_3 \cdot N_2 \cdot N_1' \cdot N_0
 \end{aligned}$$



S. Jay yang, CE, RIT

## Algebraic simplification

- Recall (T8)  $X \cdot Y + X \cdot Y' = X$

$$F = \sum_{N_3, N_2, N_1, N_0} (1, 3, 5, 7, 2, 11, 13)$$

$$= N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0 + N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0 + \dots$$

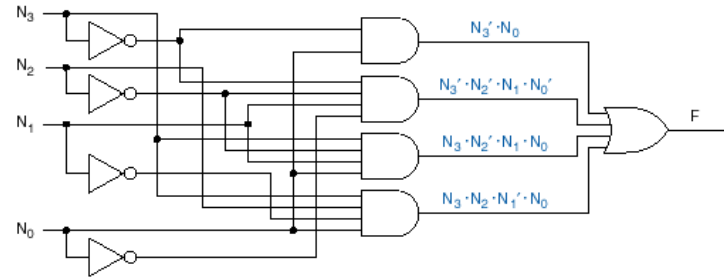
$$= (N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0) + (N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0) + \dots$$

$$= N_3' \cdot N_2' \cdot N_0 + N_3' \cdot N_2 \cdot N_0 + \dots$$

- Simplify equation to reduce number of gates & gate inputs

S. Jay Yang, CE, RIT

## Resulting circuit



S. Jay Yang, CE, RIT

## Combinational circuit design/minimization

- Objective:
  - Minimizing # logic gates
  - Minimizing # inputs to the logic gates
- Note different logic gates may have different # transistors
- General idea: simplify the Boolean expression using the theorems, especially (T10, T10', T13, T13')
- Karnaugh-map (K-map)
  - Graphical representation of the truth table
  - Offers visualization of (T10, T10')
  - Works for functions with less than 6 variables
- Real world: use programs to minimize logic circuits
  - E.g., VHDL, Verilog, ABEL, ...

S. Jay Yang, CE, RIT

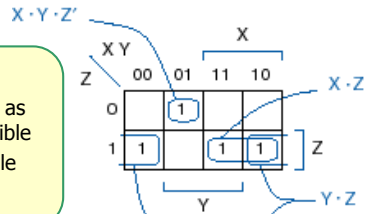
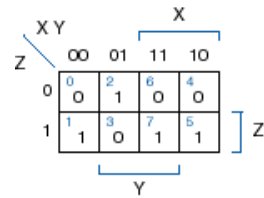
## Karnaugh-map usage

- Plot 1s corresponding to minterms of function.
- Circle largest possible rectangular sets of 1s.
  - # of 1s in set must be power of 2
  - OK to cross edges
- Read off product terms, one per circled set.
  - Variable is 1 → include variable
  - Variable is 0 → include complement of variable
  - Variable is both 0 and 1 → variable not included
- Circled sets and corresponding product terms are called 'prime implicants'
- Minimum number of gates and gate inputs

S. Jay Yang, CE, RIT

### 3 variable example: $F = \Sigma(1,2,5,7)$

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



#### Rules of thumb:

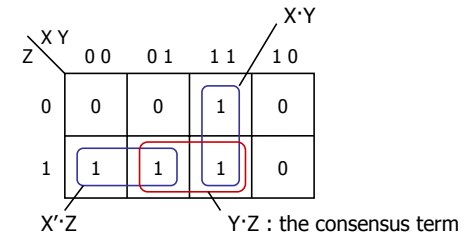
- Group (prime implicant) as large (many 1s) as possible
- As few groups as possible
- Overlaps are OK

S. Jay Yang, CE, RIT

### K-map = visualization of the theorems

#### Example: (T11) Consensus

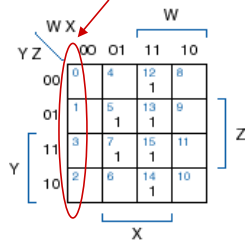
$$X \cdot Y + X' \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z$$



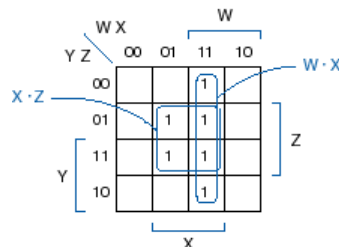
S. Jay Yang, CE, RIT

### 4 variable K-map example

Note how it maps to the rows of the truth table



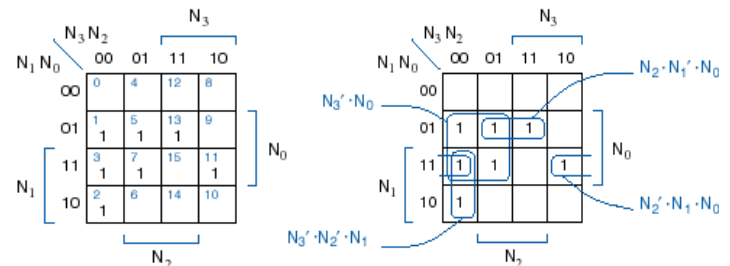
$$F = \Sigma_{W,X,Y,Z}(5,7,12,13,14,15)$$



$$F = X \cdot Z + W \cdot X$$

S. Jay Yang, CE, RIT

### Prime-number detector revisited

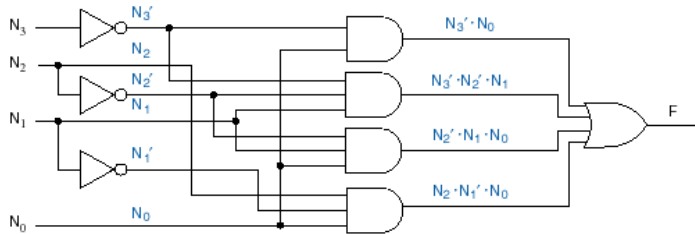


$$F = \Sigma_{N_3,N_2,N_1,N_0}(1,2,3,5,7,11,13) \quad F = N_3' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 + N_2' \cdot N_1 \cdot N_0 + N_2 \cdot N_1' \cdot N_0$$

S. Jay Yang, CE, RIT

## Compare with the previous circuit

- When we solved algebraically, we missed one simplification- the circuit below has three less gate inputs.



S. Jay Yang, CE, RIT

## K-map with don't-cares

- In some cases, the output of a combinational circuit doesn't matter for certain input combinations.
- Such combinations are called don't-cares and the output is represented in the truth table and K-maps as 'd'.
- When using K-maps to minimize such functions:
  - Allow d's to be included when grouping sets of 1's to make the sets as large as possible.
  - Do not circle any set that only contains d's.

S. Jay Yang, CE, RIT

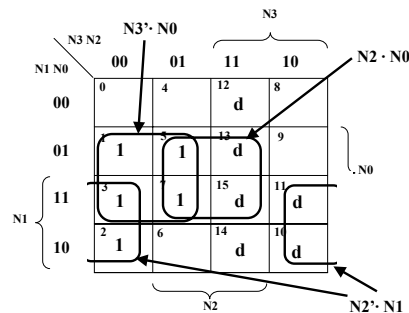
## Example with don't-cares

- Prime number detection for BCD numbers (takes value between 0-9) – minterms 10-15 are treated as don't-cares:

$$F(N_3, N_2, N_1, N_0) = \sum_{N_3, N_2, N_1, N_0} (1, 2, 3, 5, 7) + d(10, 11, 12, 13, 14, 15)$$

- Don't care: can be either 1 or 0, depending on whether large prime implicants can be formed

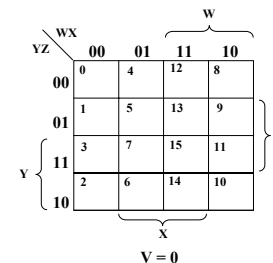
$$F = N_3' \cdot N_0 + N_2' \cdot N_1$$



S. Jay Yang, CE, RIT

## 5-variable K-maps

- The K-map for a 5-variable logic function is organized as two 4-variable K-maps:
  - Can be visualised as being one 4-variable map on top of another 4-variable map

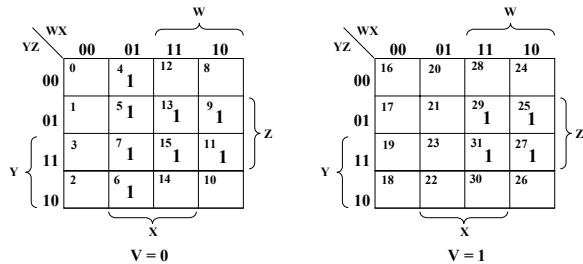


S. Jay Yang, CE, RIT

## 5-variable K-map example

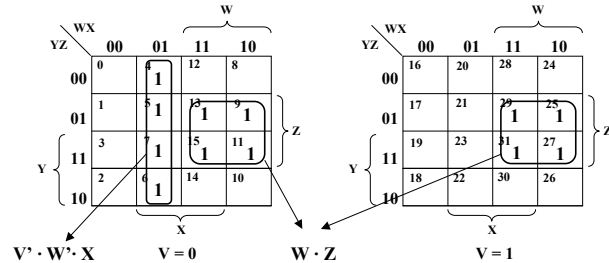
■  $F(V,W,X,Y,Z)$

$= \sum_{V,W,X,Y,Z} (4,5,6,7,9,11,13,15,25,27,29,31)$



S. Jay Yang, CE, RIT

## 5-variable K-map example – cont.



Minimum SOP:  $F = V' \cdot W' \cdot X + W \cdot Z$

S. Jay Yang, CE, RIT

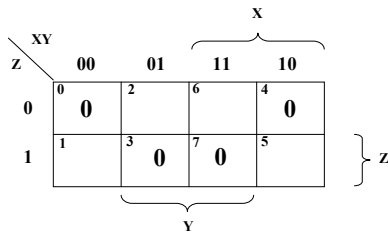
## K-map product-of-sum minimization

- Using K-map, find a minimal PoS expression for

$F(X,Y,Z) = \prod_{X,Y,Z} (0,3,4,7)$

Truth Table

Row	X	Y	Z	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

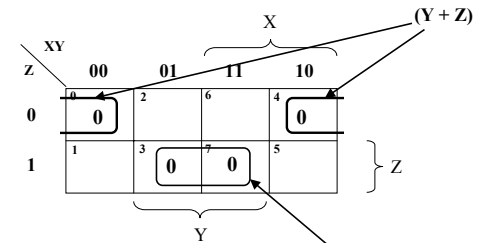


S. Jay Yang, CE, RIT

## K-map PoS minimization – cont.

Truth Table

Row	X	Y	Z	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

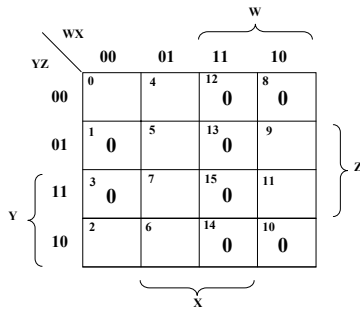


Minimum PoS:  $F = (Y + Z) \cdot (Y' + Z')$

S. Jay Yang, CE, RIT

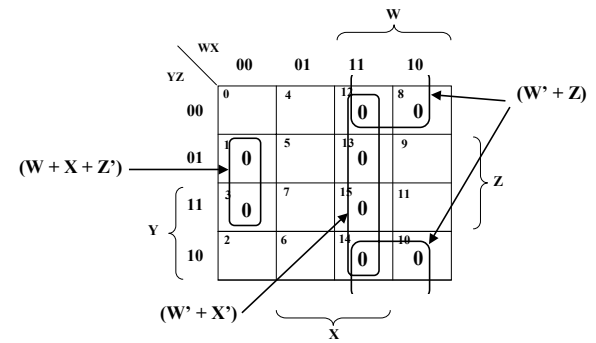
## K-map PoS minimization – another example

- Using K-map, find a minimal POS expression for
- $F(W,X,Y,Z) = \Pi_{w,x,y,z} (1,3,8,10,12,13,14,15)$



S. Jay Yang, CE, RIT

## K-map PoS minimization – another example

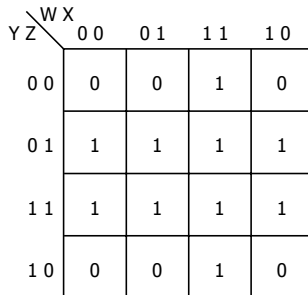


Minimum POS:  $F = (W + X + Z)' \cdot (W' + Z) \cdot (W' + X')$

S. Jay Yang, CE, RIT

## Q: Minimizing SoP or PoS via K-map?

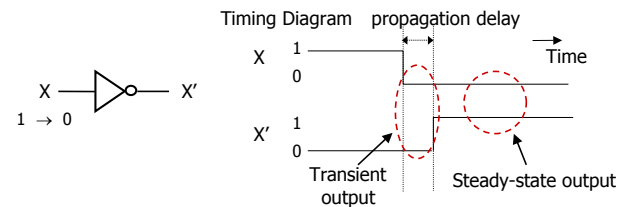
- Shall we use SoP or PoS minimization?



- How about we have 0s, 1s, and Don't-cares?

S. Jay Yang, CE, RIT

## Combinational Circuit: Transient vs. Steady-state Output

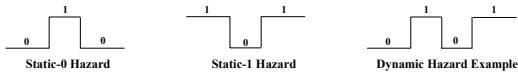


- Transient output: the temporary output due to the gate propagation delay(s)
  - Gate propagation delay: the time it takes to pull up (or down) the output signals due to the change at the input – depends on the transistor level implementation.

S. Jay Yang, CE, RIT

## Hazards in combinational circuits

- Output glitch: a momentary (transient) fluctuation in output signal due to changes in input signal.

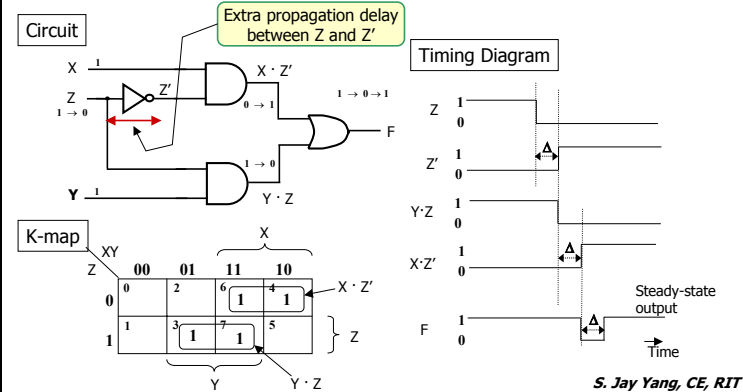


- Static hazards:
  - Static-0 hazard: The output should be 0 but goes momentarily to 1 as a result of an input change – possible in OR-AND circuits
  - Static-1 hazard: The output should be 1 but goes momentarily to 0 as a result of an input change – possible in AND-OR circuits
- Dynamic hazards: The output changes more than once as a result of a single input change (impossible in 2-level circuits).

S. Jay Yang, CE, RIT

## Example: static-1 hazard

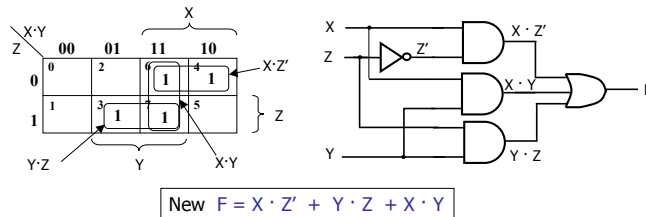
- A static-1 hazard exists in the following AND-OR circuit when  $X=1, Y=1$  and  $Z$  changes from 1 to 0 (assume all gates have propagation delay  $\Delta$ )



S. Jay Yang, CE, RIT

## Eliminate static-1 hazard using K-map

- Static-1 hazards are found using k-maps by finding adjacent 1 cells that are covered by different product terms.
- To eliminate static-1 hazards, additional product terms (prime implicants) are needed to cover such cells thus covering the transition of the variable causing the hazard.
- For the previous example the static-1 hazard is eliminated by including the additional product term  $X \cdot Y$



S. Jay Yang, CE, RIT

## Eliminate static-0 hazard using K-map

- A static-0 hazard occurs in OR-AND circuits when an input variable and its complement are connected to two different OR gates.
- The procedure to find and eliminate static-0 hazards using K-maps is done in a dual way to finding static-1 hazards.
- Static-0 hazards are found using K-maps by finding adjacent 0 cells that are covered by different sum terms.
- To eliminate static-0 hazards, additional sum terms (prime implicants) are needed to cover such cells thus covering the transition of the variable causing the hazard.

S. Jay Yang, CE, RIT