SEE3243
Digital System

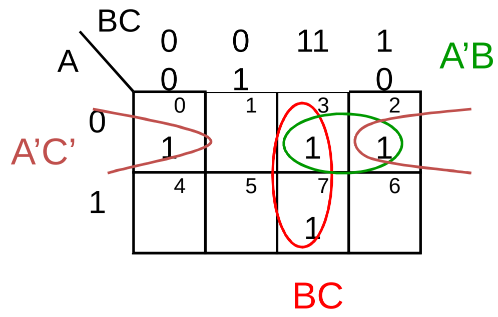
Lecturers :
Muhammad Mun'im Ahmad Zabidi
Muhammad Nadzir Marsono
Kamal Khalil

Chapter 2: Logic Theory

Implicant, Prime Implicant and Essential Prime Implicant

- **Implicant** - a single minterm or group of minterms that can be combined together on the K-map. **A'B'C', A'BC', A'BC, ABC, A'C', A'B,** and **BC**.
- **Prime Implicant** - Implicant that can not be combined with another one to remove a literal. **A'C'**, **A'B**, **BC**.
- **Essential Prime Implicant** - A prime implicant that includes a minterm not covered by any other prime implicant. **A'C'** and **BC.** Why? A'B'C' only covered by A'C' and ABC only covered by BC.

F(A,B,C) = Σm(0,2,3,7)



Finding Minimum Expression

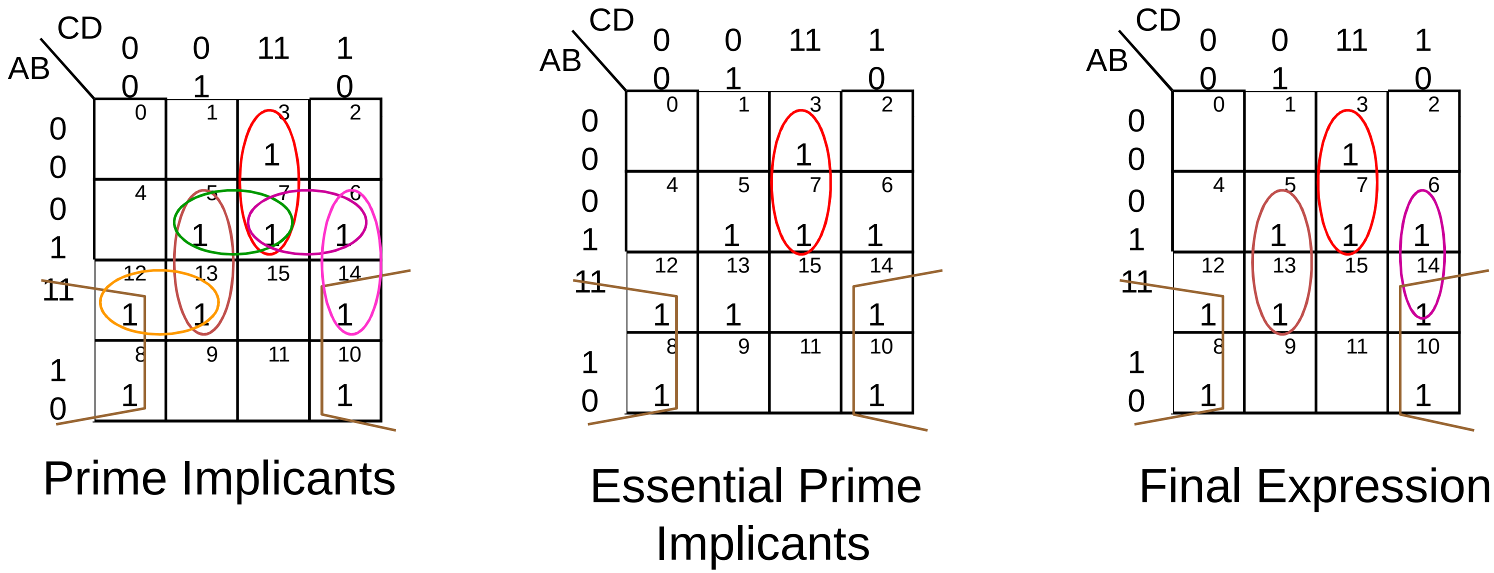
- Draw the K-map and put 1’s in each square that corresponds to a minterm of the function.
- Find the ***prime implicants***. Groups must be a power of 2.
- Find the ***essential prime implicants***. An essential prime implicant is a prime implicant that includes 1’s that are not covered by any other prime implicants.
- Write down the minimized expression. First write down all essential prime implicants. If there are any 1's not covered by prime implicants, *carefully* select prime implicants to cover the remaining 1's. Note, you may have to try several selections to find the minimal form of the expression.

Confuse with the steps? Another example

- Use a K-map to simplify the following Boolean function:

  $F(A,B,C,D) = \Pi M(0,1,2,4,9,11,15)$

  $F(A,B,C,D) = \Sigma m(3,5,6,7,8,10,12,13,14)$



F(A,B,C,D) = BC'D + AD' + A'CD + BCD'

Entered Variable K-Map (EVM)

- EVM extends map for function with too many variable for a ordinary Karnaugh Map
- Let say ordinary K Map has ***n*** map variable. So far we've looked to 4 map variables, max.
- If a function has ***(m+n)*** variables, to be fit into n-bit K-map, ***m***-bit must reside into ***n***-bit minterm in K map as ***entered variable.***
- Fortunately, in this lecture, ***m*** = 1
- Confuse?

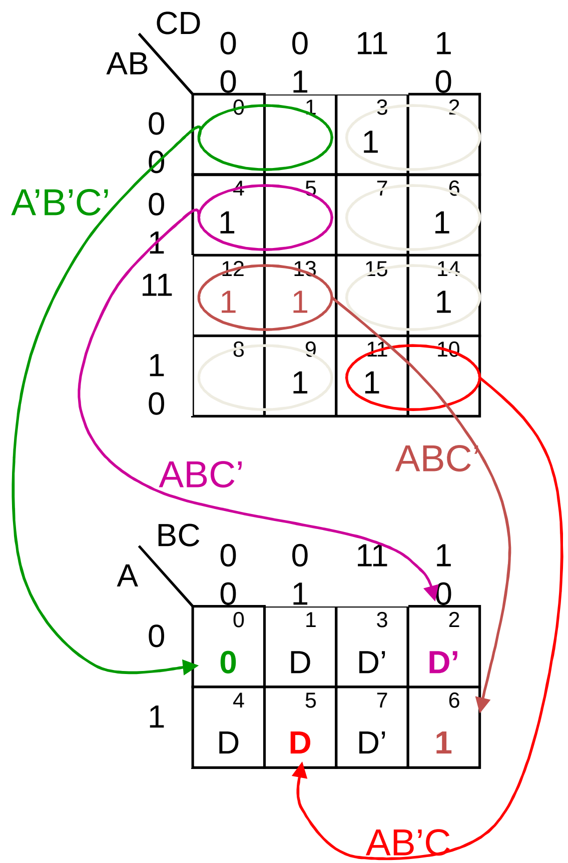
An example

- F = A’BCD + A’B’CDE + ABC’D’E’
  + A’BC’DE’
- A,B,C,D - ***map variables***.
- E - ***entered variable***. The entered variable completes the expression represented by the map variables.
- The second term is loaded in the 0011 cell because of A’B’CD. The cell is loaded with E to complete the term.
- The final 2 terms are treated in the same way. For ABC’D’E’, the 1100 cell is loaded with E. For A’BC’DE’ the 0101 cell is loaded with E’.

AB \ CD	00	01	11	10
00	0	1	3 E	2
01	4	5 E’	7 1	6
11	12 E’	13	15	14
10	8	9	11	10

SOP example

- In this example, a function with 4 variables need to be re-organise in K-map with 3 map variables only.
- In the above K-map, in every combination of A,B and C, we will compare the output with variable D.
- If the output is the complement of D, put D' inside that particular cell.
- Another way is by using truth table first, then into 3-bit K map.



SOP example

- From previous function, let say F, we first organise it into a truth table.
- Then find the entered variables.
- Reorganise into 3-bit K map.

AB \ CD	00	01	11	10
00	0	1	3 1	2
01	4 1	5	7	6 1
11	12 1	13 1	15	14 1
10	8	9 1	11 1	10

A	B	C	D	F	EV
0	0	0	0	0	0
0	0	0	1	0	
0	0	1	0	0	D
0	0	1	1	1	
0	1	0	0	1	D'
0	1	0	1	0	
0	1	1	0	1	D'
0	1	1	1	0	
1	0	0	0	0	D
1	0	0	1	1	
1	0	1	0	0	D
1	0	1	1	1	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	1	D'
1	1	1	1	0	

A \ BC	00	01	11	10
0	0 0	1 D	3 D'	2 D'
1	4 D	5 D	7 D'	6 1

How to minimise EVM

EVM grouping rules

- Entered variables can be grouped with
  - Other identical entered variables
  - 1s
- Find and circle all the single EV’s which cannot be grouped
- Find and group all single EV’s which can be grouped in only one way with other entity
- Find and group all single EV’s which can be grouped in more than one way. Group these variables
  - first with another identical ungrouped entered variable
  - second with uncovered or partially covered 1
  - **make arbitrary choice when needed**

Example 1

- Still the same function.
- ‘1’ in ABC’ can be decomposed to D+D’.
- Then, group the identical entered variables.
- Follow the procedure.
- F(A,B,C,D) = BD’ + B’CD + AC’D

A \ BC	00	01	11	10
0	0 0	1 D	3 D’	2 D’
1	4 D	5 D	7 D’	6 1

A \ BC	00	01	11	10
0	0 0	1 D	3 D’	2 D’
1	4 D	5 D	7 D’	6 D’+D

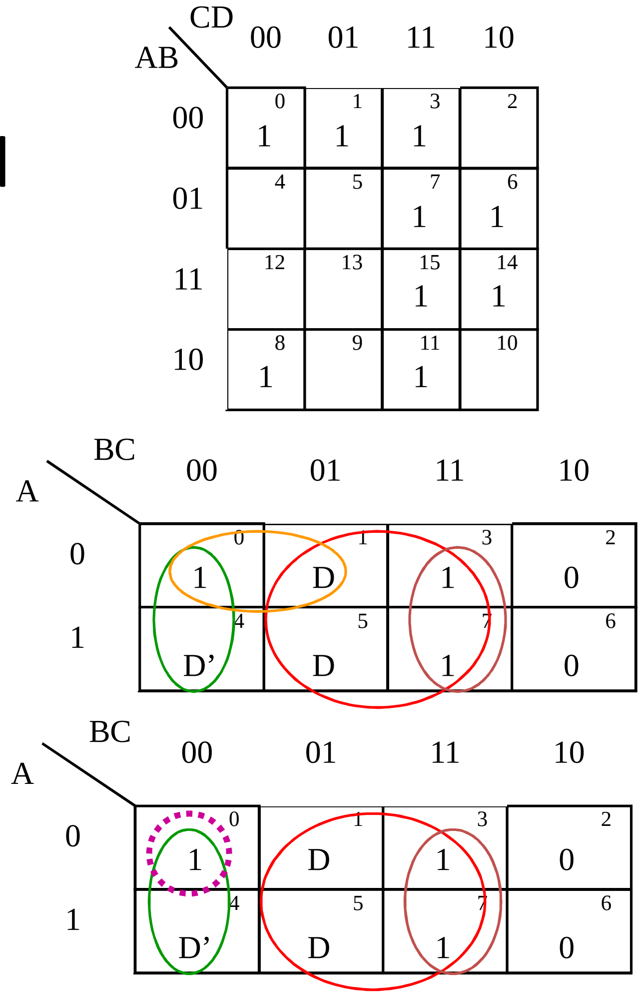
- Now, compare function F from EVM in previous slide with the minimisation result from the original 4-bit K map.
- Do you get the same answer?

AB \ CD	00	01	11	10
00	0	1	3 1	2
01	4 1	5	7	6 1
11	12 1	13 1	15	14 1
10	8	9 1	11 1	10

F(A,B,C,D) =______________________________

Example 2

- Lets call this function G.
  - '1' in A'B'C' can be decomposed to D+D'.
  - Then, group the identical entered variables.



- Follow the procedure.
  - F(A,B,C,D) = CD + B'C'D' + BC + A'B'D

- Now, compare function G from EVM in previous slide with the minimisation result from the original 4-bit K map.
- Do they yield the same answer?

AB \ CD	00	01	11	10
00	0 1	1 1	3 1	2
01	4	5	7 1	6 1
11	12	13	15 1	14 1
10	8 1	9	11 1	10

G(A,B,C,D) = ______________________________

Example 3

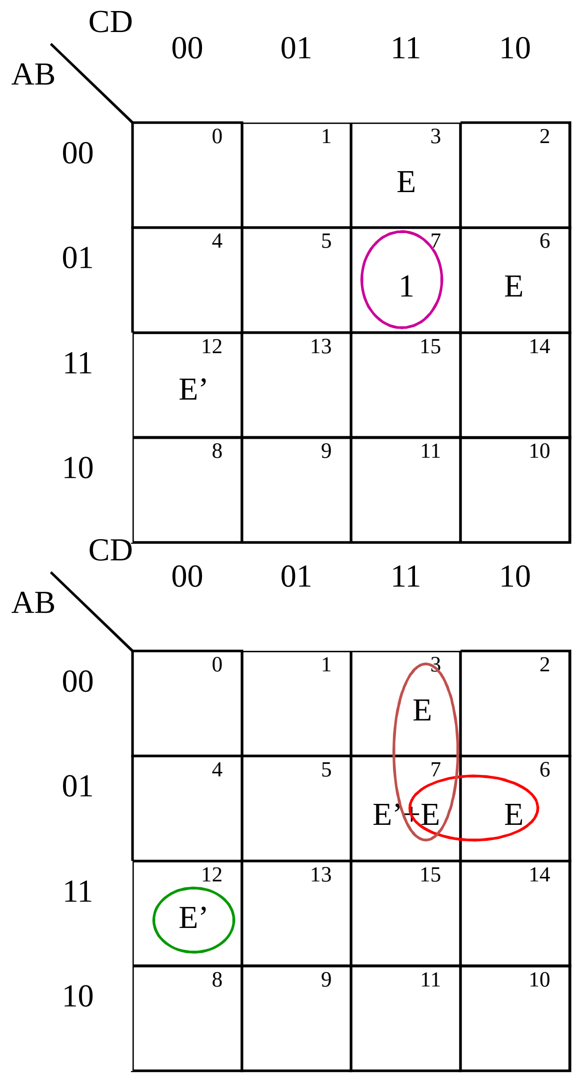
- Change 1 to E+E’
- Then group entries with neighbour cells that have the same entered variable
- Note that all the entities are covered
- F = ABC’D’E’ + A’BDE’ + A’CDE

AB \ CD	00	01	11	10
00	0	1	3 E	2
01	4	5 E’	7 1	6
11	12 E’	13	15	14
10	8	9	11	10

AB \ CD	00	01	11	10
00	0	1	3 E	2
01	4	5 E’	7 E’+E	6
11	12 E’	13	15	14
10	8	9	11	10

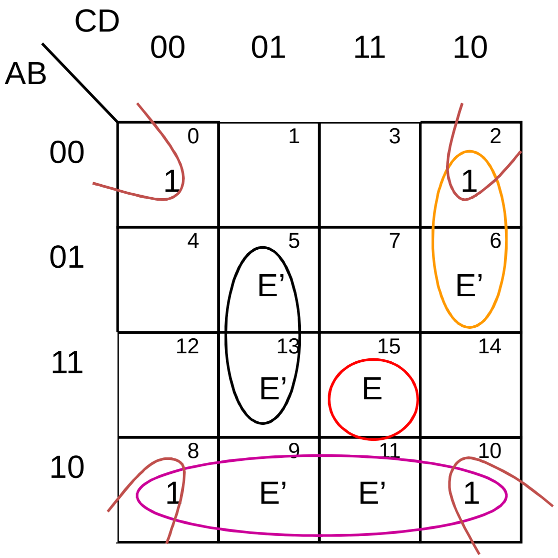
Example 4

- What if not all entities covered?
- In this example, partially covered '1' at A'BCD must be covered (so that the E' variable is covered).
- Thus, F = ABC'D'E' + A'BCE + A'CDE + A'BCD



Example 5

- F is a function of A,B,C,D,E.
- Let E become entered variable



F(A,B,C,D,E)
= B'D' + AB'E' + BC'DE' +
A'CD'E' + ABCDE

A	B	C	D	E	**F**		A	B	C	D	E	**F**
0	0	0	0	0	**1**		1	0	0	0	0	**1**
0	0	0	0	1	**1**		1	0	0	0	1	**1**
0	0	0	1	0	**0**		1	0	0	1	0	**1**
0	0	0	1	1	**0**		1	0	0	1	1	**0**
0	0	1	0	0	**1**		1	0	1	0	0	**1**
0	0	1	0	1	**1**		1	0	1	0	1	**1**
0	0	1	1	0	**0**		1	0	1	1	0	**1**
0	0	1	1	1	**0**		1	0	1	1	1	**0**
0	1	0	0	0	**0**		1	1	0	0	0	**0**
0	1	0	0	1	**0**		1	1	0	0	1	**0**
0	1	0	1	0	**1**		1	1	0	1	0	**1**
0	1	0	1	1	**0**		1	1	0	1	1	**0**
0	1	1	0	0	**1**		1	1	1	0	0	**0**
0	1	1	0	1	**0**		1	1	1	0	1	**0**
0	1	1	1	0	**0**		1	1	1	1	0	**0**
0	1	1	1	1	**0**		1	1	1	1	1	**1**

How to solve EVM in POS?

- Almost the same procedure like solving EVM in SOP.
- The differences,
  - Group identical EV
  - Substitute ‘0’ with EV.EV’ (Example D.D’).
  - Except for 0-0 grouping, complement EV when writing in SOP.
- Complement using De-Morgan
- More confused? You should.

Example 1

- Lets call this function H.
- Note that in cell A'B'C', the EV is '0', and written as D'.D.

AB \ CD	00	01	11	10
00	$_{0}$ 0	$_{1}$ 0	$_{3}$	$_{2}$ 0
01	$_{4}$	$_{5}$ 0	$_{7}$ 0	$_{6}$
11	$_{12}$	$_{13}$	$_{15}$ 0	$_{14}$
10	$_{8}$ 0	$_{9}$	$_{11}$	$_{10}$ 0

A \ BC	00	01	11	10
0	$_{0}$ D'.D	$_{1}$ D	$_{3}$ D'	$_{2}$ D'
1	$_{4}$ D	$_{5}$ D	$_{7}$ D'	$_{6}$ 1

$$H'(A,B,C,D) = B'D' + A'C'D + BCD$$

$$H(A,B,C,D) = (B+D)(A+C+D')(B'+C'+D')$$

- Now, compare function H from EVM in previous slide with the minimisation result from the original 4-bit K map.
- Do they yield the same answer?

AB \ CD	00	01	11	10
00	0: 0	1: 0	3:	2: 0
01	4:	5: 0	7: 0	6:
11	12:	13:	15: 0	14:
10	8: 0	9:	11:	10: 0

H’(A,B,C,D) = ____________________

H(A,B,C,D) = ____________________

DIY

- Now, you try.
- Check your answer with 4-bit K map POS minimisation.

AB \ CD	00	01	11	10
00	0	1	3	2: 0
01	4: 0	5: 0	7	6
11	12: 0	13: 0	15	14
10	8	9: 0	11	10: 0

EVM

H’(A,B,C,D) = ____________________

H(A,B,C,D) = ____________________

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

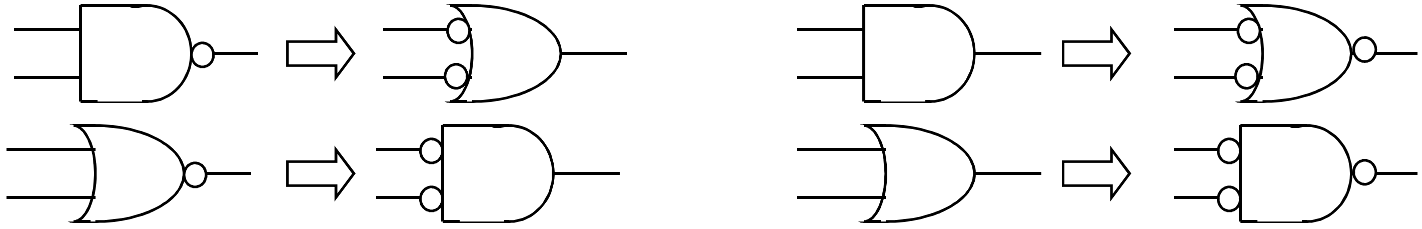
4-bit K map

H’(A,B,C,D) = ____________________________

H(A,B,C,D) = ____________________________

Logic Transformation

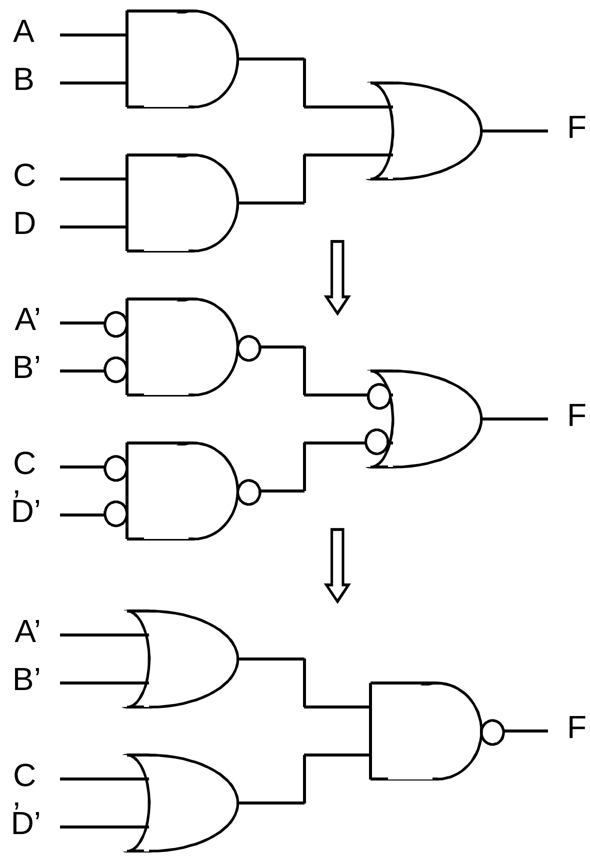
- By performing transformation operations, different (and possibly simpler) gate networks for the same function are possible.
- Simple equivalencies:



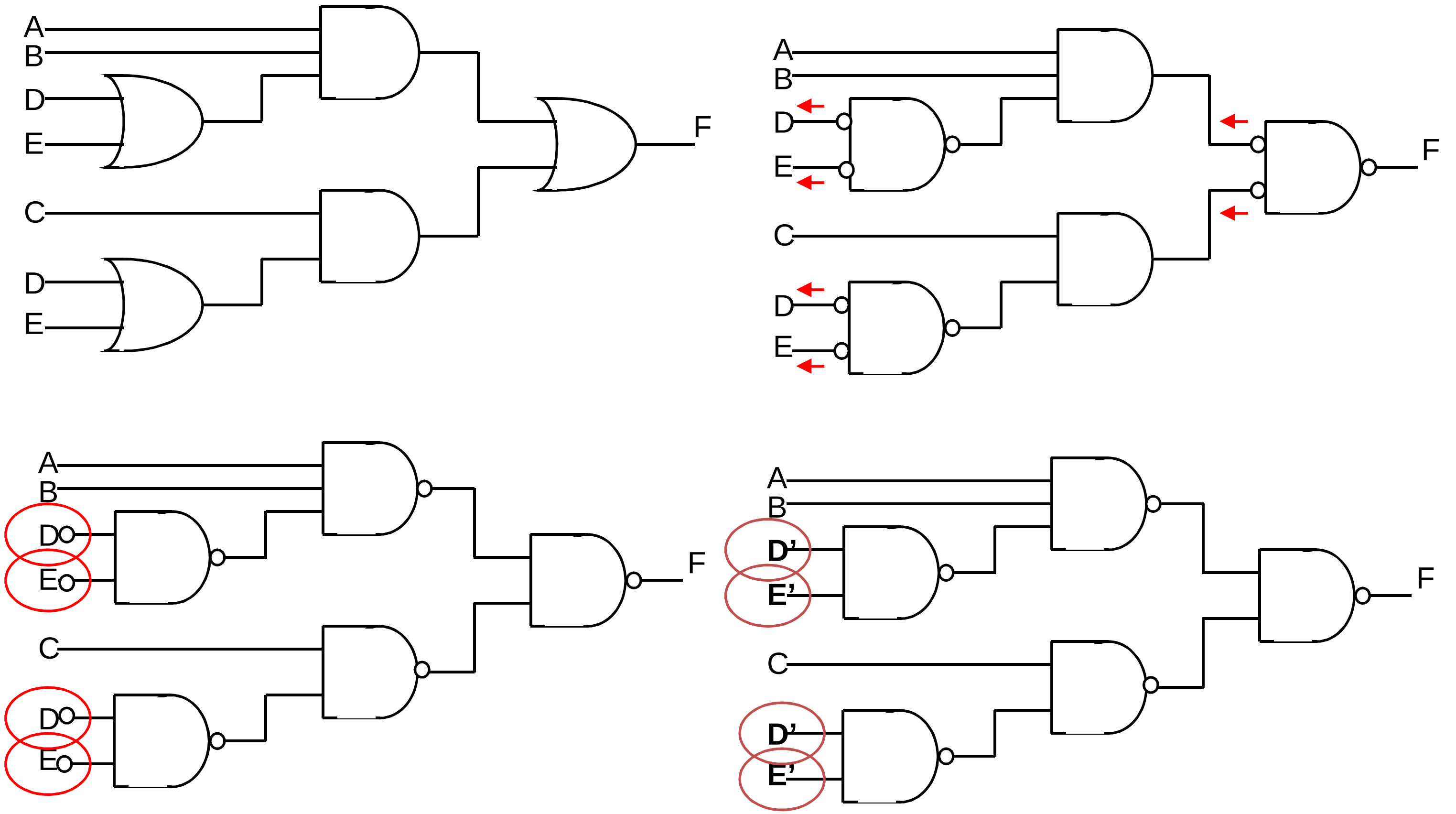
- Based on DeMorgan's Theorems
  - (AB)'=A'+B'
  - (A+B)'=A'B'
- Thus, AND-OR networks can be transformed to OR-NAND, NAND-NAND, NOR-OR, etc.

Example

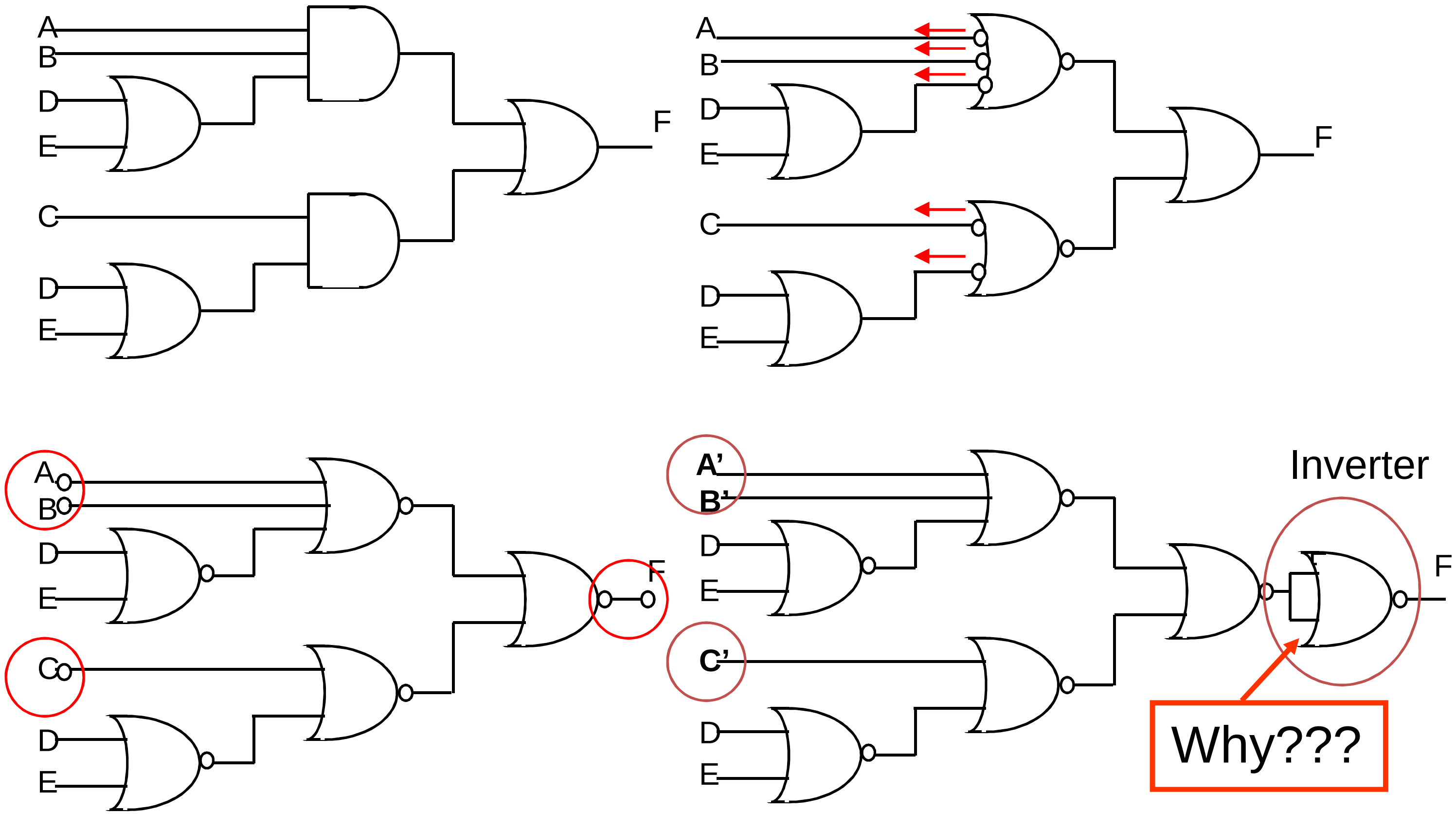
- This transformation changes AND-OR form into OR-NAND form.
- However, usually we are interested in transforming circuit (SOP or POS) into negative logics (yes.. NAND, NOR, Inverter).
- Why? Negative logic is extensively used in VLSI.



Another Example (NAND-NAND)



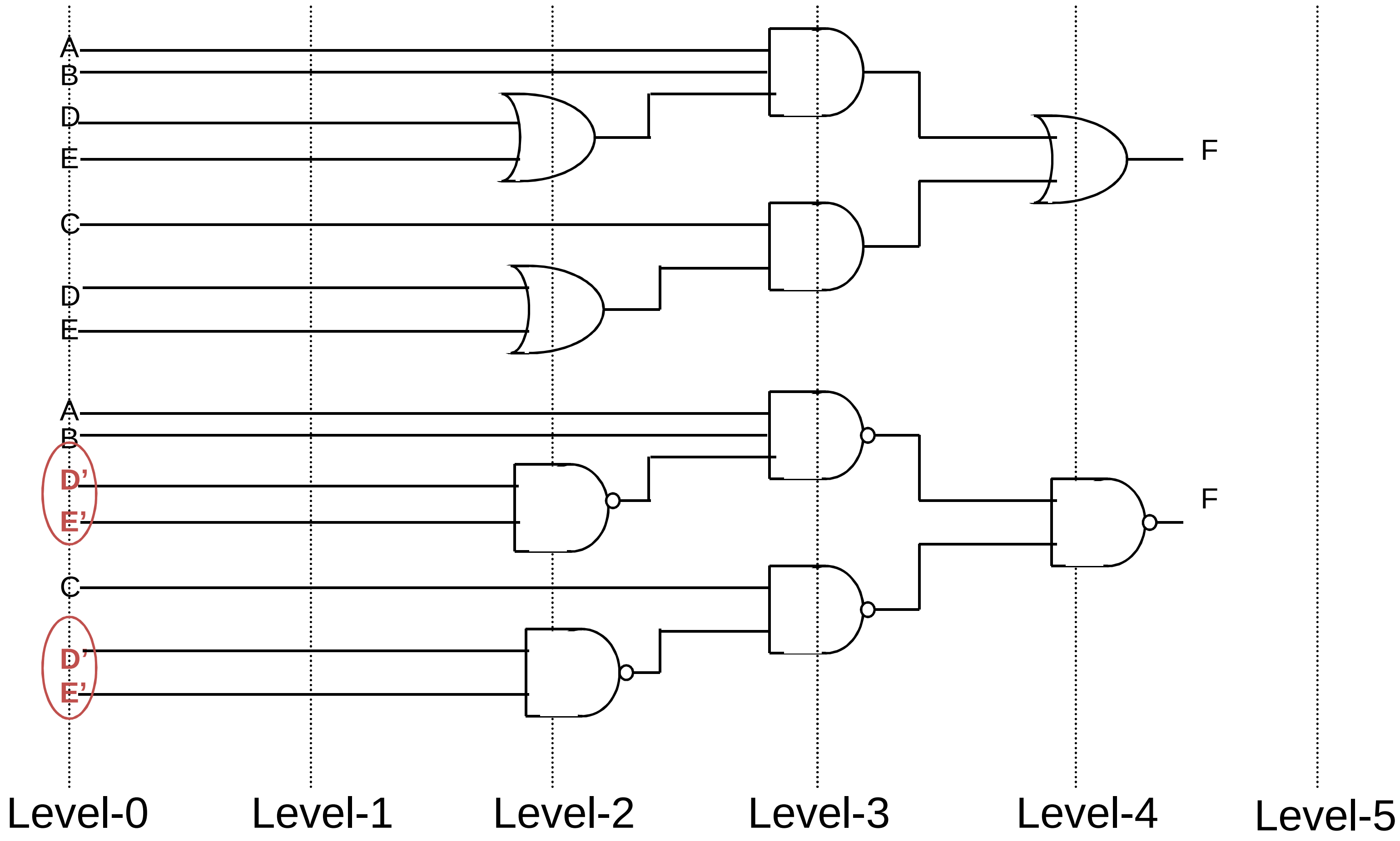
Another Example (NOR-NOR)



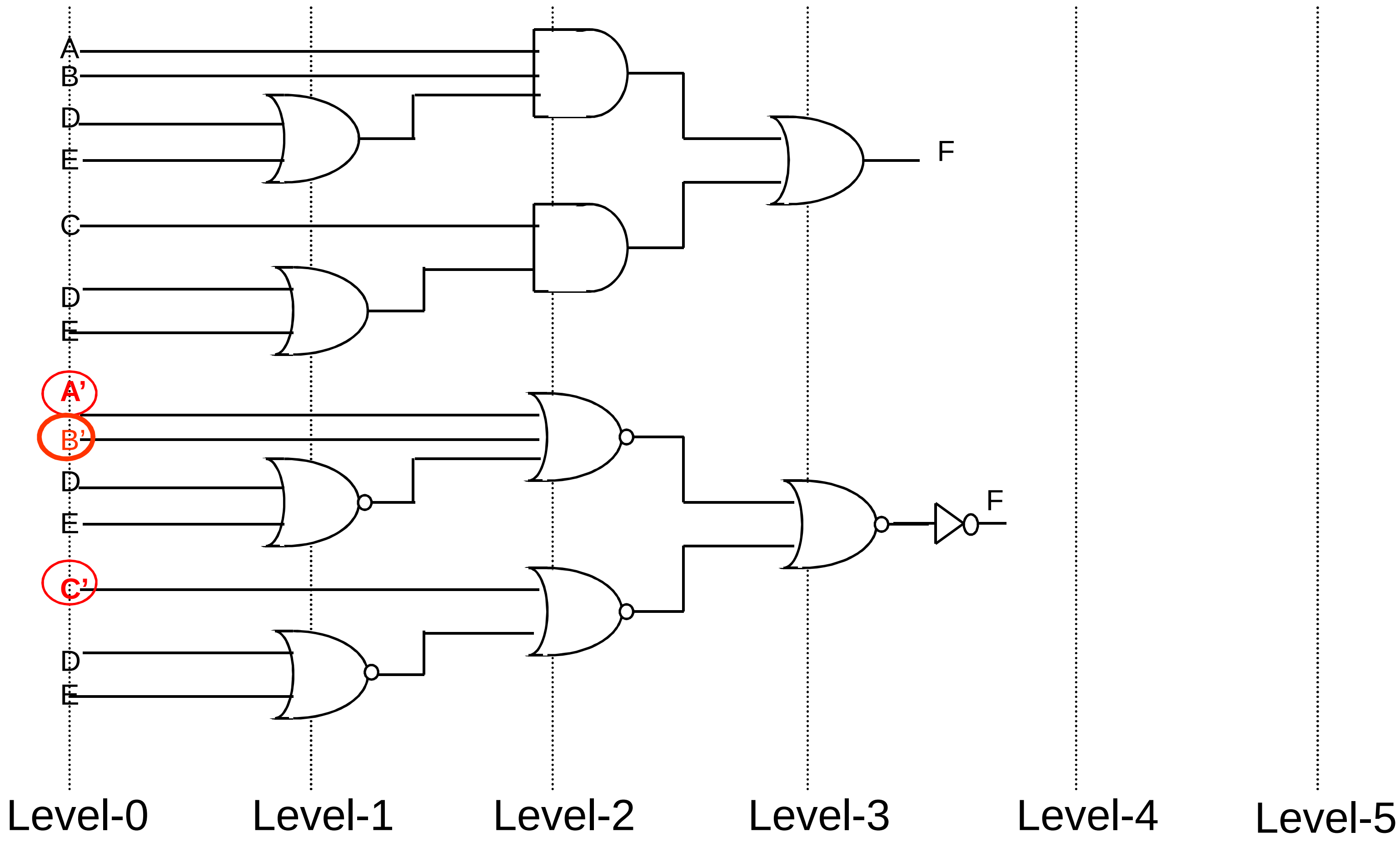
Transformation: General Rules (AND-OR-INVERTER Network only)

- To convert to NAND-only form
  - Begin with SOP form with an OR gate output (if not, add inverter at the output ***later***)
  - AND (odd level) and OR (even level) gates must alternate between levels! (1,2,..n)
  - Level 0 for input
  - Replace all gates with NAND gates
  - Put inverter if no gate between 2***k*** (k = 1,2,..m) level.
- Algorithm for NOR-form is similar, but start with POS form, OR (odd level) and AND (even level). Still, if the output is not AND gate, add inverter later.

Example 1 (NAND-NAND)



Example 2 (NOR-NOR)



DIY 1

- Implement this function using NAND-NAND transformation (don't derive the K-map or the truth table)

$$F(A,B,C,D,E,F,G,H,I,J) = ((A+B)+C+DE).((G+H).F.I).(J)$$

DIY 2

- Implement this function using NOR-NOR transformation
- $F(A,B,C,D) = (B+D)(A+C+D')(B'+C'+D')$

DIY3

- Transform this equation into NOR-NOR gates
  - $F(A,B,C,D) = (AB'(C'+BD'))+A'(BC+D(A'+B))$

DIY 4

- Transform this equation into NAND-NAND gates
  - $F(A,B,C,D) = (AB'(C'+BD'))+A'(BC+D(A'+B))$

Introduction to Hazard and Glitch

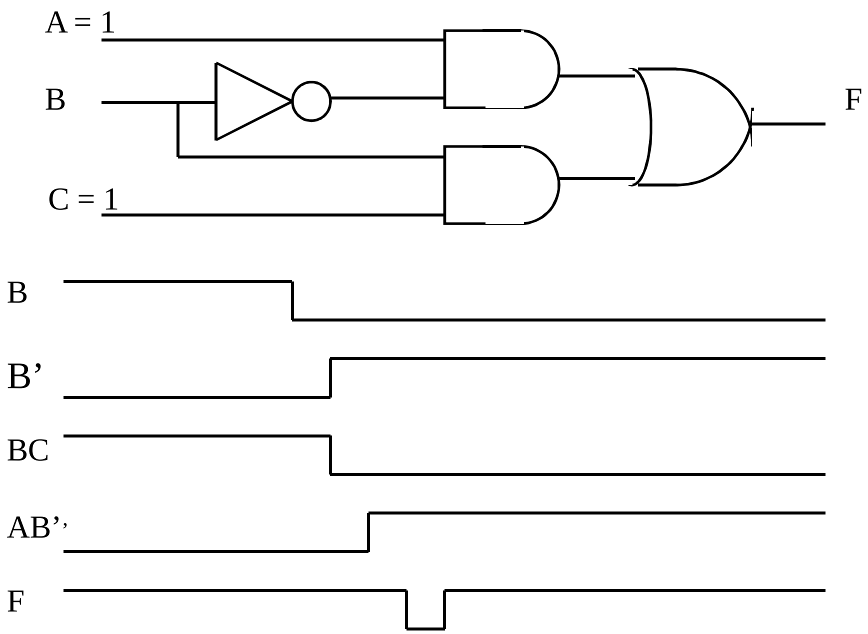
- A ***hazard*** is a characteristic of a circuit.
- A ***glitch*** is an unwanted pulse that may occur in a circuit with a hazard.
- A circuit with a hazard may or may not glitch.
- As an ***analogy*** a wet floor at the supermarket is a hazard. It's not a problem unless someone slips and falls which would be a glitch.
- The state of a circuit and the pattern of input values determines if a circuit with a hazard will cause glitch.

Types of Hazards

- A static hazard is when the output should remain static but experiences an unwanted pulse.
- A dynamic hazard is when the output should go through a smooth transition but changes more than once before settling at the new value.

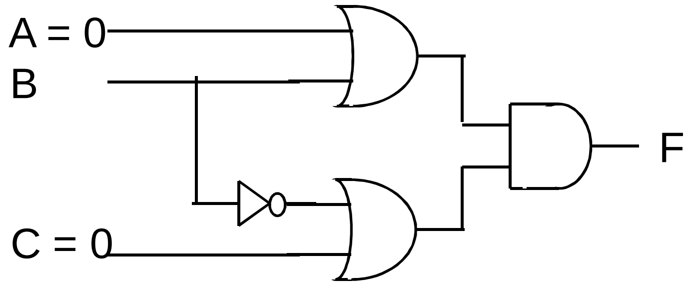
Example 1

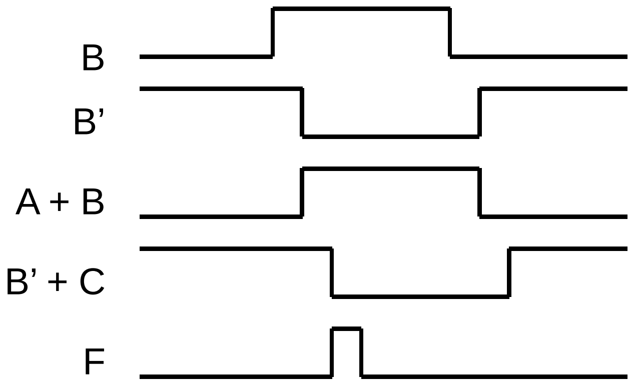
- F(A,B,C) = AB’ + BC



Example 2

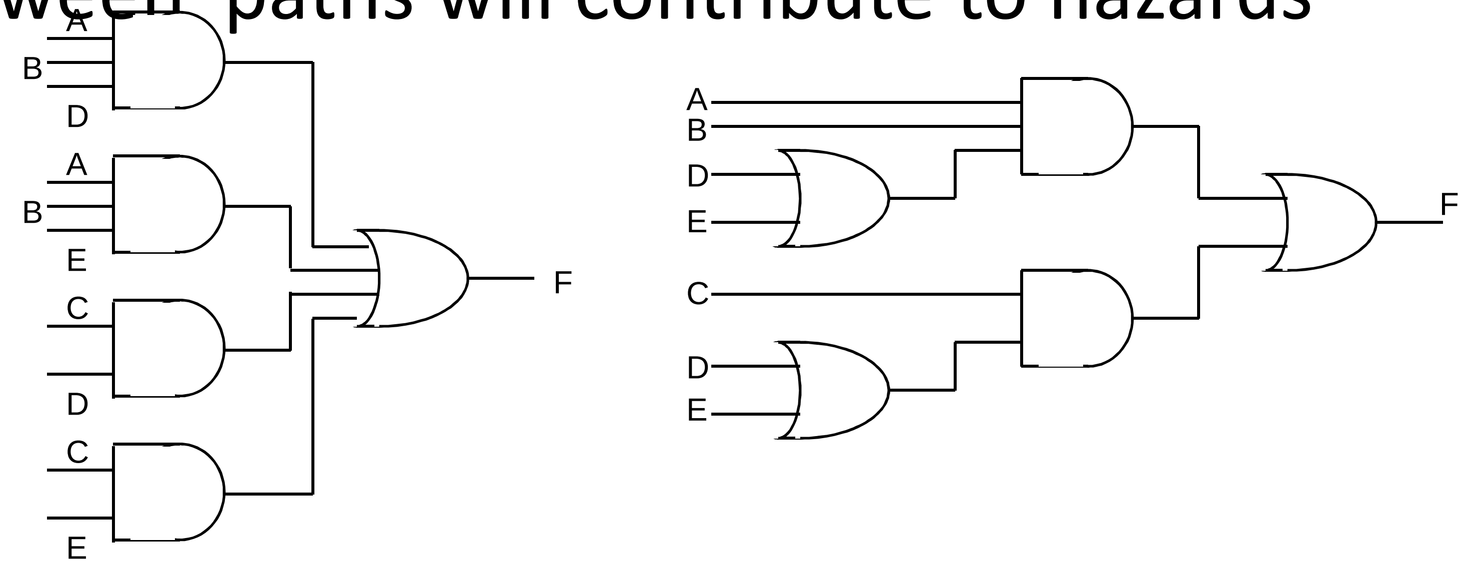
- $F(A,B,C) = (A + B)(B' + C)$





Example 3

- F = ABD+ABE+CD+CE
- 5 gates, 14 inputs
- It can be factored: simpler but has short and long delay paths. Inequalities of delays between paths will contribute to hazards



What's Next

- You will look at the MSI design. How to use standard devices to realise digital circuitry.
- Make sure you understand this chapter first.