Sistemas Digitais I

LESI - 2º ano

Lesson 4 - Combinational Systems Principles

Prof. João Miguel Fernandes
(miguel@di.uminho.pt)

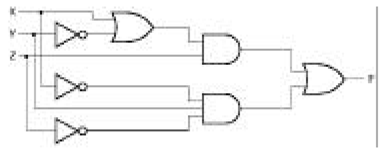
Dept. Informática

UNIVERSIDADE DO MINHO
ESCOLA DE ENGENHARIA

4. Combinational Principles

- Circuit Analysis (1) -

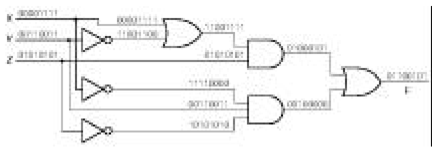
- After having a formal description of the logic function, we can:
  - **Determine the behaviour of the circuit for various input combinations.**
  - **Manipulate the algebraic description to suggest different circuit structures.**
  - **Transform the algebraic description into a standard form (example: PLD).**
  - **Use the algebraic description of the circuit to analyse a larger system that includes it as a subsystem.**



4. Combinational Principles

- Circuit Analysis (2) -

- We can obtain the truth table by going through all input combinations.



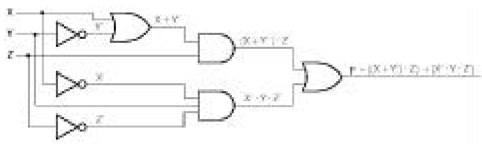
Row	X	Y	Z	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

- From the truth table we can directly write a logic expression.
- This technique is time-consuming and only works for small number of input variables.

4. Combinational Principles

- Circuit Analysis (3) -

- From inputs to outputs

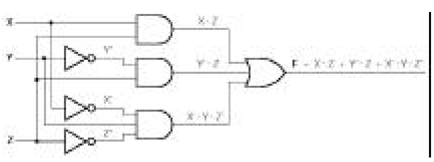


- $F = ((X+Y') \cdot Z) + (X' \cdot Y \cdot Z')$

4. Combinational Principles

- Circuit Analysis (4) -

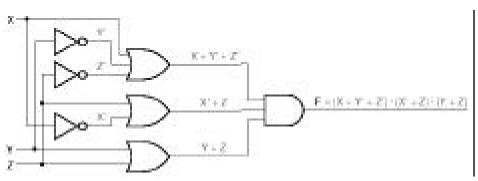
- After some algebraic transformation
- $F = ((X+Y') \cdot Z) + (X' \cdot Y \cdot Z')$
  $= (X \cdot Z) + (Y' \cdot Z) + (X' \cdot Y \cdot Z')$



4. Combinational Principles

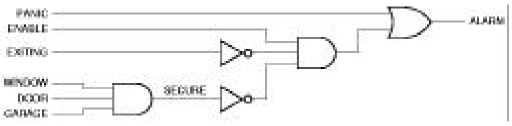
- Circuit Analysis (5) -

- $F = ((X+Y') \cdot Z) + (X' \cdot Y \cdot Z')$
  $= (X+Y'+X') \cdot (X+Y'+Y) \cdot (X+Y'+Z') \cdot (Z+X') \cdot (Z+Y) \cdot (Z+Z')$
  $= 1 \cdot 1 \cdot (X+Y'+Z') \cdot (Z+X') \cdot (Z+Y) \cdot 1 =$
  $= (X+Y'+Z') \cdot (X'+Z) \cdot (Y+Z)$



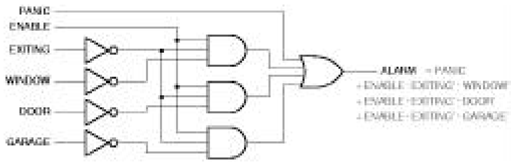
4. Combinational Principles
- Circuit Synthesis (1) -

- The starting point for designing combinational circuits is usually a description written in natural language (Portuguese, for example).
- Example: Construction of an alarm circuit.
  "The ALARM output is 1 if the PANIC input is 1, or if the ENABLE input is 1, the EXITING input is 0, and the house is not secure. The house is secure if the WINDOW, DOOR and GARAGE inputs are all 1".
- ALARM = PANIC + ENABLE·EXITING'·SECURE'
  SECURE = WINDOW·DOOR·GARAGE



4. Combinational Principles
- Circuit Synthesis (2) -

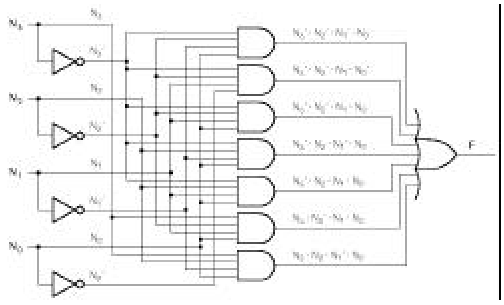
- **ALARM = PANIC + ENABLE·EXITING'·SECURE'**
  SECURE = WINDOW·DOOR·GARAGE
- **ALARM = PANIC + ENABLE·EXITING'·(WINDOW·DOOR·GARAGE)'**
- **ALARM = PANIC + ENABLE·EXITING'·(WINDOW'+DOOR'+GARAGE')**
- **ALARM = PANIC + ENABLE·EXITING'·WINDOW' + ENABLE·EXITING'·DOOR' + ENABLE·EXITING'·GARAGE'**



4. Combinational Principles
- Circuit Synthesis (3) -

- Other times, the description starts with a list of the input combinations for which a signal should be on or off (equivalent to a truth table).
- Example: Construction of a circuit that detects 4-bit prime numbers.
  "Given a 4-bit input combination $N=N_3N_2N_1N_0$, the circuit produces a 1 output for N=1,2,3,5,7,11,13."
- $F = \sum_{N_3,N_2,N_1,N_0} (1,2,3,5,7,11,13) =$
  $N_3'\cdot N_2'\cdot N_1'\cdot N_0 + N_3'\cdot N_2'\cdot N_1\cdot N_0' + N_3'\cdot N_2'\cdot N_1\cdot N_0 + N_3'\cdot N_2\cdot N_1'\cdot N_0 +$
  $N_3'\cdot N_2\cdot N_1\cdot N_0 + N_3\cdot N_2'\cdot N_1\cdot N_0 + N_3\cdot N_2\cdot N_1'\cdot N_0$

4. Combinational Principles
- Circuit Synthesis (4) -

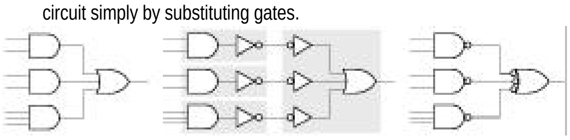


4. Combinational Principles
- Circuit Manipulations (1) -

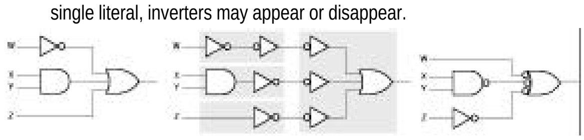
- We have described methods that use AND, OR, and NOT gates.
- In some situations, we might like to use NAND or NOR gates (they are faster than ANDs and ORs in most technologies).
- However, most people don´t develop logic propositions in terms of NAND and NOR connectives.
- Nobody says: "I don´t like a girl, if she is not smart or not pretty and also if she is not rich or not friendly". [G' = (S'+P') · (R'+F')]
- It is more common to say: "I like a girl, if she is smart and pretty or if she is rich and friendly". [G = (S·P) + (R·F)]
- Any logic expression can be transformed into an equivalent sum-of-products (SOP) expression and implement with AND and OR gates.

4. Combinational Principles
- Circuit Manipulations (2) -

- A 2-level AND-OR circuit may be converted to a 2-level NAND-NAND circuit simply by substituting gates.



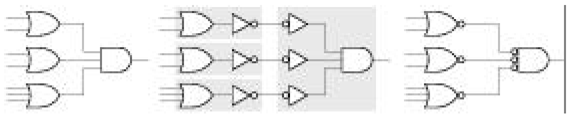
- If any product terms in the sum-of-products expression contain just a single literal, inverters may appear or disappear.



4. Combinational Principles

- Circuit Manipulations (3) -

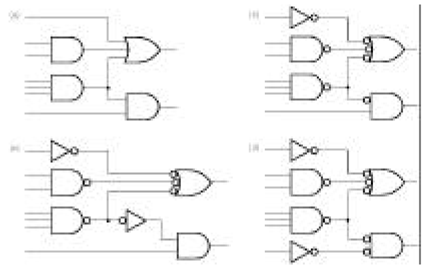
- Any sum-of-products (SOP) expression can be realised in either two ways (AND-OR circuit or NAND-NAND circuit).
- The principle of duality can be applied to this rule:
 Any product-of-sums (POS) expression can be realised in either two ways (OR-AND circuit or NOR-NOR circuit).



4. Combinational Principles

- Circuit Manipulations (4) -

- These manipulations can be applied to arbitrary logic circuits.

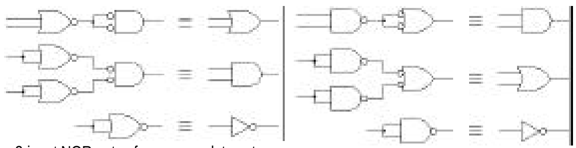


- The (d) solution is better than the (c) solution.

4. Combinational Principles

- Circuit Manipulations (5) -

- Any set of logic-gate types that can realize any logic function is called a <u>complete set</u>.
- 2-input AND gates, 2-input OR gates and inverters form a complete set.
- Any logic function can be expressed as a sum-of-products of variables and their complements, and AND and OR gates with any number of inputs can be made from 2-input gates.



2-input NOR gates form a complete set.

2-input NAND gates form a complete set.

4. Combinational Principles

- Minimisation (1) -

- It is uneconomical to realise a logic function directly from the first expression that comes up.
- Canonical (sum and product) expressions are especially expensive.
- Logic minimisation uses several techniques to obtain the simplest gate-level implementation of a Boolean function.
- But simplicity depends on the metric used.
- Three possible metrics that can be used are:
  - number of literals
  - number of gates
  - number of cascaded levels of gates

4. Combinational Principles

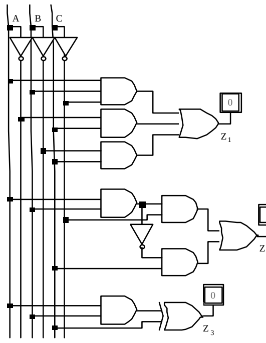
- Minimisation (2) -

- The <u>number of literals</u> measure the amount of wiring needed to implement a function.
- The <u>number of gates</u> measures circuit area.
- There is a relation between the number of gates in a design and the number of components needed for its implementation.
- The <u>number of levels of gates</u> is related with the circuit's delay.
- Reducing the number of levels reduces overall delay.
- However, putting a circuit in a form suitable for minimum delay rarely yields an implementation with the fewest or simplest gates.
- It is not possible to minimise all three metrics at the same time.

4. Combinational Principles

- Minimisation (3) -

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



Two-Level Realization (inverters don't count)

Multi-Level Realization
+ Reduced Gate Fan-ins
- Bigger number of levels

Complex Gate: XOR
+ Fewest Gates
- Increased Delay

4. Combinational Principles

- Minimisation (4) -

- Minimisation techniques reduce the number and size of gates that are needed to build a circuit, thus decreasing the cost of the system.
- The minimisation methods reduce the cost of a 2-level AND-OR or OR-AND circuit by:
  - **minimising the number of first-level gates;**
  - **minimising the number of inputs on each first-level gate;**
  - **minimising the number of inputs on each second-level gate;**

4. Combinational Principles

- Minimisation (5) -

- The minimisation methods do not consider the cost of input inverters.
- They assume that both true and complement versions of all input variables are available (appropriate for PLD-based design).
- They also assume that the function to be minimised is represented as a truth table or as a minterm or maxterm list.
- Minimisation is based on theorems T10 and T10':
  product·Y + product·Y' = product
  (sum+Y) · (sum+Y') = sum
- If two terms differ only in one variable, they can be combined into a single term with one less variable.
- One gate is saved and the remaining one has one fewer input.

4. Combinational Principles

- Minimisation (6) -

A	B	F
0	0	0
0	1	0
1	0	1
1	1	1

$F = A\,B' + A\,B = A\,(B' + B) = A$

B's values change within the on-set rows

B is eliminated, A remains

A's values don't change within the on-set rows

A	B	G
0	0	1
0	1	0
1	0	1
1	1	0

$G = A'\,B' + A\,B' = (A' + A)\,B' = B'$

B's values stay the same within the on-set rows

A is eliminated, B remains

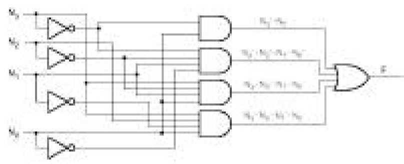
A's values change within the on-set rows

Essence of Simplification:
Find two element subsets of the ON-set where only one variable changes its value. This single varying variable can be eliminated!

4. Combinational Principles

- Minimisation (7) -

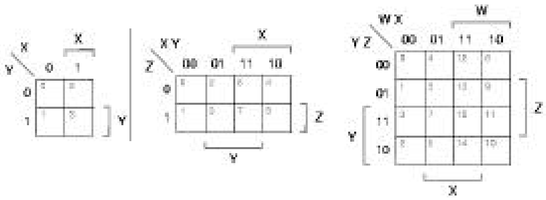
- Let us apply this technique to the prime-number detector function.
- $F = \Sigma_{N_3,N_2,N_1,N_0}(1,2,3,5,7,11,13) =$
  $N_3'{\cdot}N_2'{\cdot}N_1'{\cdot}N_0 + N_3'{\cdot}N_2'{\cdot}N_1{\cdot}N_0 + N_3'{\cdot}N_2{\cdot}N_1'{\cdot}N_0 + N_3'{\cdot}N_2{\cdot}N_1{\cdot}N_0 + \ldots =$
  $(N_3'{\cdot}N_2'{\cdot}N_1'{\cdot}N_0 + N_3'{\cdot}N_2'{\cdot}N_1{\cdot}N_0) + (N_3'{\cdot}N_2{\cdot}N_1'{\cdot}N_0 + N_3'{\cdot}N_2{\cdot}N_1{\cdot}N_0) + \ldots =$
  $(N_3'{\cdot}N_2'{\cdot}N_0) + (N_3'{\cdot}N_2{\cdot}N_0) + \ldots = N_3'{\cdot}N_0 + \ldots$



4. Combinational Principles

- Karnaugh Maps (1) -

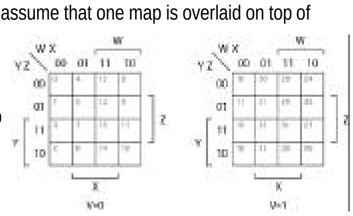
- It is hard to find terms that can be combined.
- A Karnaugh map is a graphical representation of a logic function's truth table.
- The map for an n-input logic function is an array with 2^n cells, one for each minterm.



4. Combinational Principles

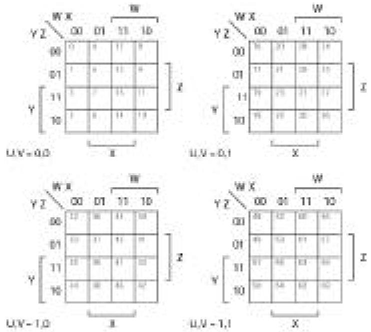
- Karnaugh Maps (2) -

- The Karnaugh maps used to represent 5- and 6-variable functions are not as convenient as 2-, 3- and 4-variable maps, since adjacency is more difficult to visualize.
- In a 5-variable map, we need to use 2 4-variable maps that are located next to each other.
- In this representation, we assume that one map is overlaid on top of the other, so as to create a 3-dimensional object.
- Each square is adjacent to 5 squares (4 on its map and 1 on the other one).



4. Combinational Principles
- Karnaugh Maps (3) -

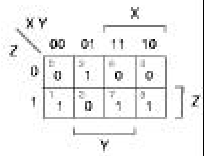
A 6-variable Karnaugh map.



4. Combinational Principles
- Karnaugh Maps (4) -

- To represent a logic function on a Karnaugh map, copy 1s and 0s from the truth table to the corresponding map's cells.
- Each map cell corresponds to a minterm of the function.

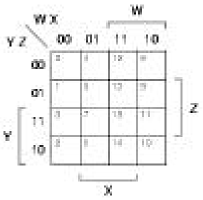
X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



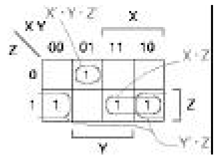
- In real life, we just copy 1s or 0s (not both) to the cells, depending on the expression we want to obtain (SOP or POS).

4. Combinational Principles
- Karnaugh Maps (5) -

- Why that strange ordering of rows/columns?
- Each cell corresponds to an input combination that differs in only one variable from each of its immediately adjacent neighbours.
- Cells 7 and 15 in the 4-variable map differ only in the value of W.
- In the 3- and 4-variable maps, cells on the left/right or top/bottom borders are also neighbours.
- Cells 8 and 10 in the 4-variable map differ only in the value of Y.
- Since pairs of adjacent 1-cells correspond to minterms that differ in one variable, they can be combined into a single product term, using theorem T10 (product·Y + product·Y' = product).

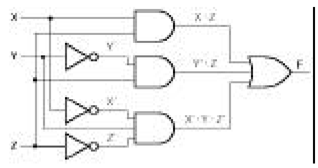


4. Combinational Principles
- Karnaugh Maps (6) -



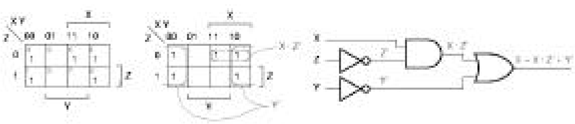
- Cells 5 and 7
  F = ... + X·Y'·Z + X·Y·Z
  = ... + X·Z
- Cells 1 and 5
  F = X'·Y'·Z + X·Y'·Z + ...
  = Y'·Z + ...

- Minterm 5 is included twice. No problems, since X+X=X.
- F = X·Z + Y·Z + X·Y·Z'



4. Combinational Principles
- Karnaugh Maps (7) -

- The cell-combining procedure can be extended to combine more than two 1-cells into a single term.
- $F = \sum_{X,Y,Z}(0,1,4,5,6) = X'\cdot Y'\cdot Z' + X'\cdot Y'\cdot Z + X\cdot Y'\cdot Z' + X\cdot Y'\cdot Z + X\cdot Y\cdot Z'$
  $= [Y'\cdot(X'\cdot Z') + Y'\cdot(X'\cdot Z) + Y'\cdot(X\cdot Z') + Y'\cdot(X\cdot Z)] + X\cdot Y\cdot Z'$
  $= Y' + X\cdot Y\cdot Z'$



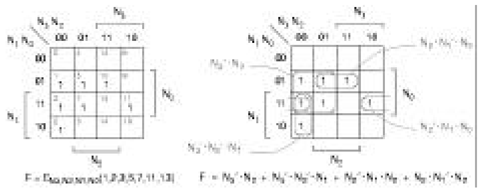
4. Combinational Principles
- Karnaugh Maps (8) -

- In general, 2^i 1-cells may be combined to form a product term containing n-i literals (n = number of variables).
- Rule for combining 1-cells:
  - **A set of 2^i 1-cells may be combined if there are i variables that take on all 2^i combinations within that set, while the remaining n-i variables have the same value throughout that set.**
  - **The respective product term has n-i literals, where a variable is complemented if it appears as 0 in all of the 1-cells, and uncomplemented if it appears as 1.**
- Graphically, we can circle rectangular sets of 2^n 1-cells.

4. Combinational Principles

- Karnaugh Maps (9) -

- From the 1-cell circles, obtain the corresponding product term:
  - **If a circle covers only areas of the map where the variable is 0 (1), then the variable is complemented (uncomplemented) in the product term.**
  - **If a circle covers areas of the map where the variable is 0 and 1, then the variable does not appear in the product term.**



4. Combinational Principles

- Karnaugh Maps (10) -

- A minimal sum of a logic function F is a sum-of-products expression for F such that no sum-of-products expression for F has fewer product terms, and any sum of products expression with the same number of product terms has at least as many literals.
- The minimal sum has the fewest possible product terms (1st level gates and 2nd level gate inputs) and the fewest possible literals (1st level gate inputs).
- A logic function P implies a logic function F ($P \Rightarrow F$) if for every input combination such that P=1, then F=1 also. F includes or covers P.
- A prime implicant of a logic function F is a normal product term P that implies F, such that if any variable is removed from P, then the resulting product term does not imply F.

4. Combinational Principles

- Karnaugh Maps (11) -

- In terms of a Karnaugh map, a prime implicant of F is a circled set of 1-cells, such that if we try to make it larger (covering twice as many cells), it covers one or more 0s.
- Prime-Implicant Theorem: A minimal sum is a sum of prime implicants.
- To find a minimal sum, we need not consider any product terms that are not prime implicants.
- The sum of all the prime implicants of a function is called a complete sum.
- The complete sum is not necessarily a minimal one.

4. Combinational Principles

- Karnaugh Maps (12) -

Algorithm: Minimum SOP Expression from a Karnaugh Map

Step 1: Choose an “1” not already covered by an implicant.
Find “maximal” groupings of 1s (and Xs) adjacent to that element.
Remember to consider top/bottom row, left/right column, and corner adjacencies. This forms a prime implicant.

Repeat Step 1 to find all prime implicants.

Step 2: Visit a “1”. If it is covered by a single prime implicant, it is essential, and participates in the final expression. The 1s covered by it do not need to be revisited.

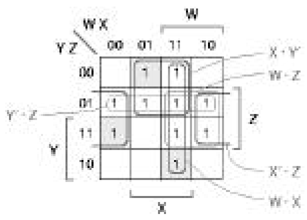
Repeat Step 2 until all essential prime implicants have beem found.

Step 3: If there remain 1s not covered by essential prime implicants, then select the smallest number of prime implicants that cover the remaining 1s.

4. Combinational Principles

- Karnaugh Maps (13) -

- $F = \Sigma_{W,X,Y,Z,}(1,3,4,5,9,11,12,13,14,15)$
- The function has 5 prime implicants.
- The minimal sum includes only 3 prime implicants:
  $F = X \cdot Y' + X' \cdot Z + W \cdot X$
- How to determine which prime implicants to include?



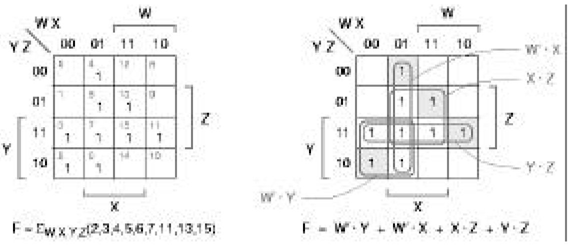
4. Combinational Principles

- Karnaugh Maps (14) -

- A distinguished 1-cell of a logic function is an input combination that is covered by only one prime implicant.
- An essential prime implicant of a logic function is a prime implicant that covers one or more distinguished 1-cells.
- Essential prime implicants must be included in every minimal sum.
- The 1st step in the prime implicant selection is identifying the distinguished 1-cells and including the corresponding prime implicants.
- Then, one needs only to determine how to cover the 1-cells, if any, that are not covered by the essential prime implicants.

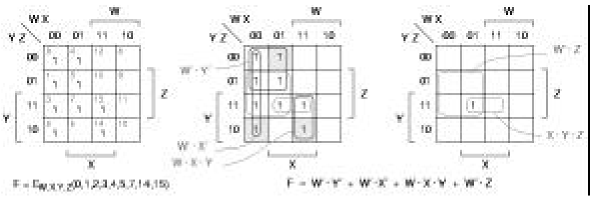
4. Combinational Principles

- Karnaugh Maps (15) -



4. Combinational Principles

- Karnaugh Maps (16) -



4. Combinational Principles

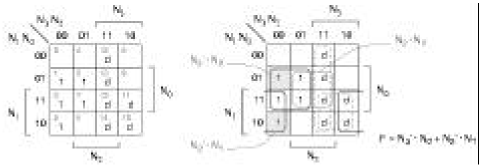
- Karnaugh Maps (17) -

- Using duality, we can minimise product-of-sums expressions by looking at 0s at the Karnaugh map.
- Each 0 on the map corresponds to a maxterm.
- An easier way to find minimal products is to find the minimal sum for F'.
- The 1s of F' are just the 0s of F.
- Once we have the minimal sum for F', we complement the result by using the generalised DeMorgan's theorem (T14), to obtain a minimal product for F.
- Example: $F' = X \cdot Y' + X' \cdot Z + W \cdot X$
  $F = (X'+Y) \cdot (X+Z') \cdot (W'+X')$

4. Combinational Principles

- Karnaugh Maps (18) -

- Sometimes the output of a function does not matter for certain input combinations, called <u>don't cares.</u>
- Example: A prime-number detector whose 4-bit input N is always a BCD digit. Minterms 10-15 should never occur.
- $F = \Sigma_{N_3,N_2,N_1,N_0} (1,2,3,5,7) + d(10,11,12,13,14,15)$



4. Combinational Principles

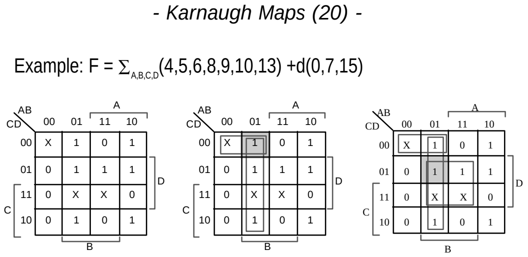
- Karnaugh Maps (19) -

- The procedure for circling sets of 1s is modified if don't cares (d's or Xs) are included.
  - **Allow d's to be included when circling sets of 1s, to make the sets as large as possible. This reduces the number of variables in the corresponding prime implicants.**
  - **Do not circle any sets that contain only d's. Including the corresponding product term in the function would unnecessarily increase its cost.**
- The remainder of the procedure is the same.
- In particular, we look for distinguished 1-cells and not for distinguished d-cells.
- We also include only the corresponding essential prime implicants, and any others that are needed to cover all the 1s on the map.

4. Combinational Principles

- Karnaugh Maps (20) -

Example: $F = \Sigma_{A,B,C,D}(4,5,6,8,9,10,13) + d(0,7,15)$

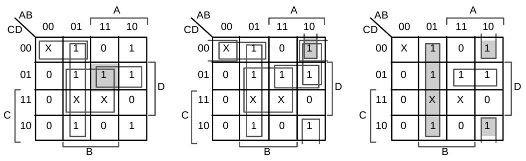


Initial Karnaugh map | **Primes around m_4=A' B C' D'** | **Primes around m_5=A' B C' D**

4. Combinational Principles

- Karnaugh Maps (21) -

Example continued



Primes around m_{13}=A B C' D'

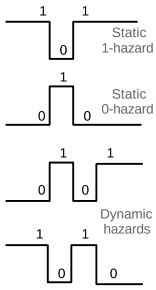
Primes around m_8=A B' C' D'

Essential primes with minimum cover

4. Combinational Principles

- Hazards (1) -

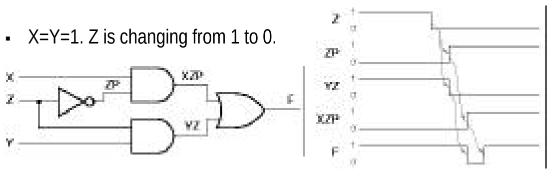
- Due to delays on electronic devices, a circuit may produce a glitch.
- A glitch is a shorth-duration change in the output value, when no change is expected.
- A hazard exists when a circuit has the possibility of producing a glitch.
- A static hazard occurs when it is possible for an output to undergo a momentary transition when it is expected to remain unchanged.
- A dynamic hazard occurs when the output has the potential to change more than once when it is expected to make a single transition.



4. Combinational Principles

- Hazards (2) -

- A static-1 hazard is a pair of input combinations that differ only in one input variable, both give 1 as output, such that a momentary 0 output may occur, during a transition in the differing input variable.
- X=Y=1. Z is changing from 1 to 0.

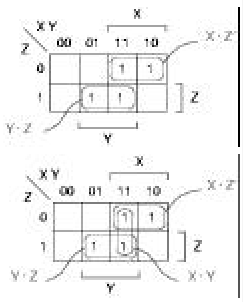


- Methods for eliminating hazards assume that only one input changes at a time. This assumption is equivalent to moving along neighbour cells in a Karnaugh map.

4. Combinational Principles

- Hazards (3) -

- Karnaugh maps can be used to detect static hazards.
- A well designed 2-level sum-of-products circuit may only have static-1 hazards.
- No single product term covers combinations XYZ=111 and XYZ=110.
- It is possible for the output to glitch momentarily to 0 (if the gate that goes to 0 changes before the gate that goes to 1).
- To eliminate the hazard, an extra AND gate must be included.

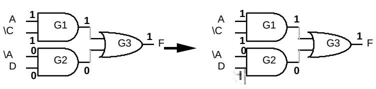


4. Combinational Principles

- Hazards (4) -

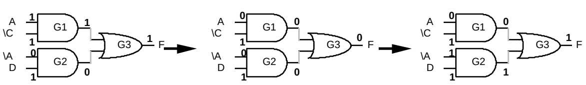
- F = $\Sigma_{A,B,C,D,}$(1,3,5,7,8,9,12,13)

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



ABCD = 1100

ABCD = 1101



ABCD = 1101

ABCD = 0101 ($\overline{A}$ is still 0)

ABCD = 0101 ($\overline{A}$ is 1)