Sistemas Digitais I
LESI :: 2º ano

Conceitos sobre Sistemas Combinacionais

António Joaquim Esteves
João Miguel Fernandes

www.di.uminho.pt/~aje

**Bibliografia: secções 4.2, 4.3 e 4.5, DDPP, Wakerly
2.1.3, 2.3.1, 2.3.5 e 3.4, CLD, Katz**

DEP. DE INFORMÁTICA
ESCOLA DE ENGENHARIA
UNIVERSIDADE DO MINHO

3. Conceitos sobre Sistemas Combinacionais

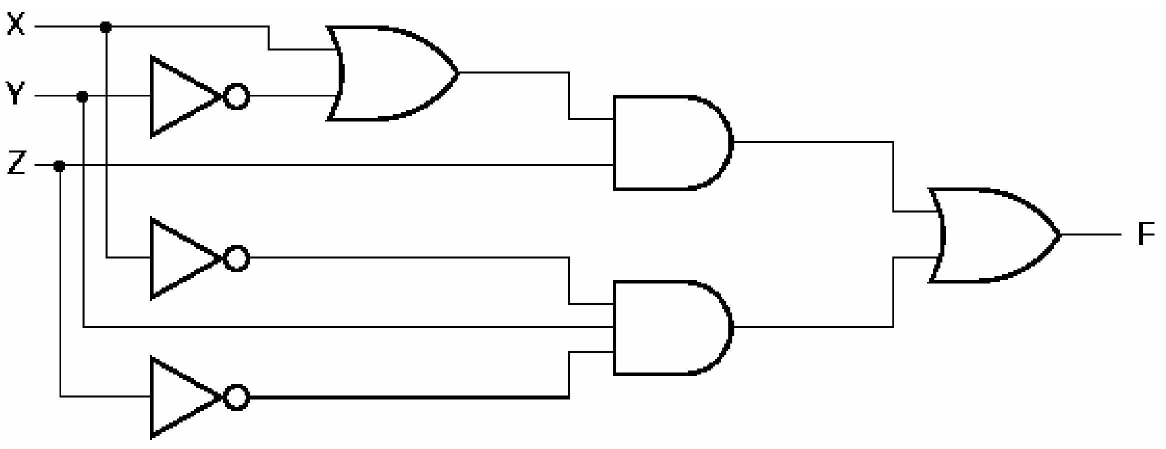
- Sumário -

- *Análise, síntese e alterações a um circuito*
- Minimização
- Mapas de Karnaugh
- *Hazards*

3. Conceitos sobre Sistemas Combinacionais

- Análise dum Circuito (1) -

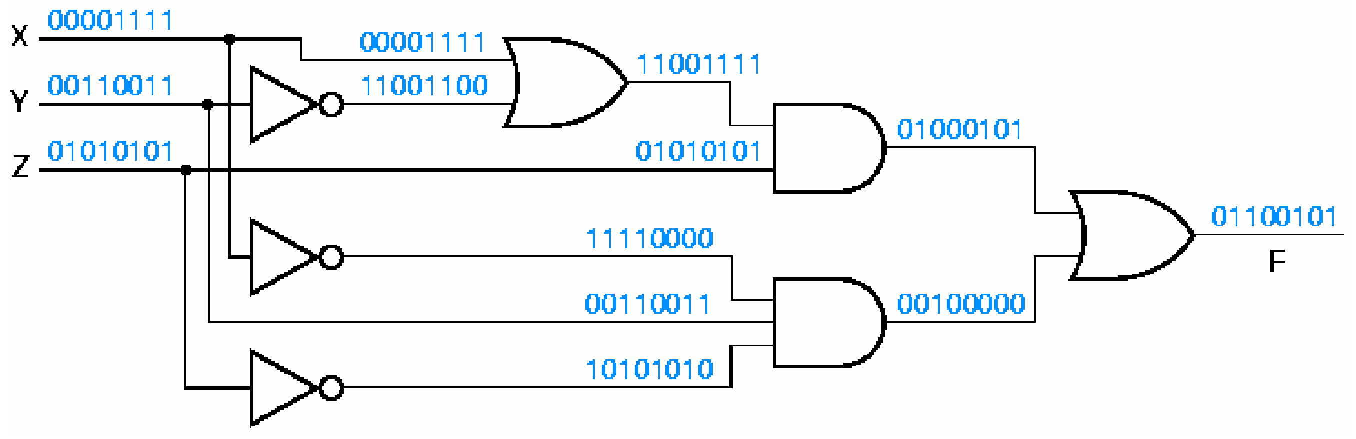
- Após obter a descrição formal da função lógica dum circuito, pode:
  - Determinar-se o seu comportamento para várias combinações de entradas.
  - Alterar-se a descrição algébrica de modo a induzir diferentes estruturas para o circuito.
  - Transformar a descrição algébrica numa forma normalizada (exemplo: PLD).
  - Utilizar a descrição algébrica do circuito para analisar um sistema de maior dimensão, do qual este sistema seja parte integrante.



3. Conceitos sobre Sistemas Combinacionais

- Análise dum Circuito (2) -

- Pode obter-se a <u>tabela de verdade</u> do circuito inspeccionando o seu comportamento para todas as combinações (2^n) das entradas.



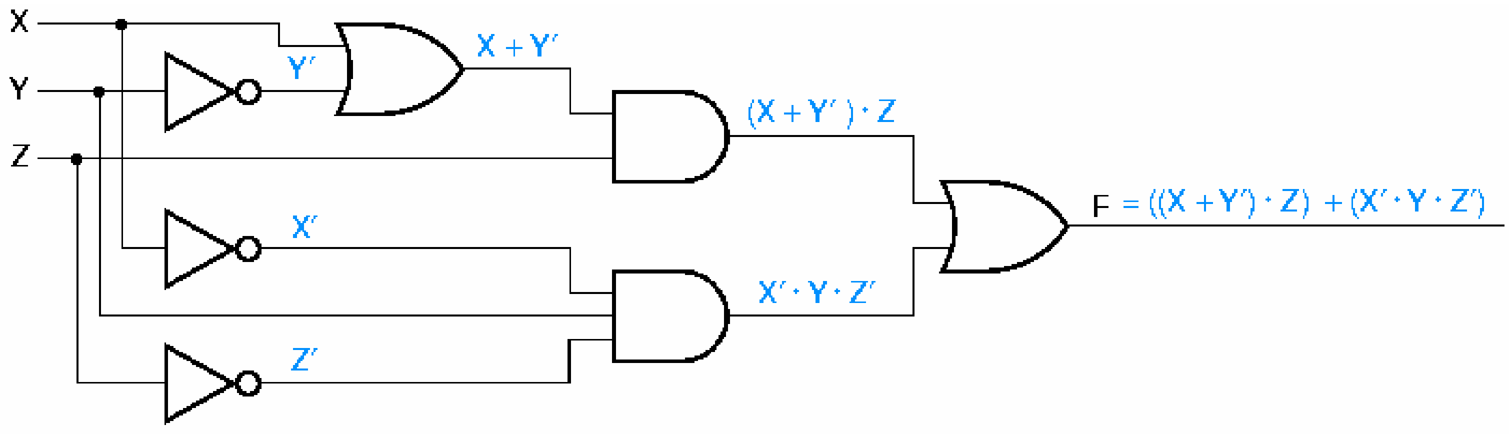
Row	X	Y	Z	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

- A partir da tabela de verdade pode extrair-se de forma directa uma expressão lógica (por ex.,SOP).
- Esta técnica é muito morosa e só é exequível se o número de variáveis de entrada é reduzido → <u>abordagem algébrica</u>.

3. Conceitos sobre Sistemas Combinacionais

- Análise dum Circuito (3) -

- <u>Abordagem algébrica:</u> analisa-se o circuito desde as entradas até às saídas, construindo uma expressão correspondente aos operadores e à estrutura do circuito



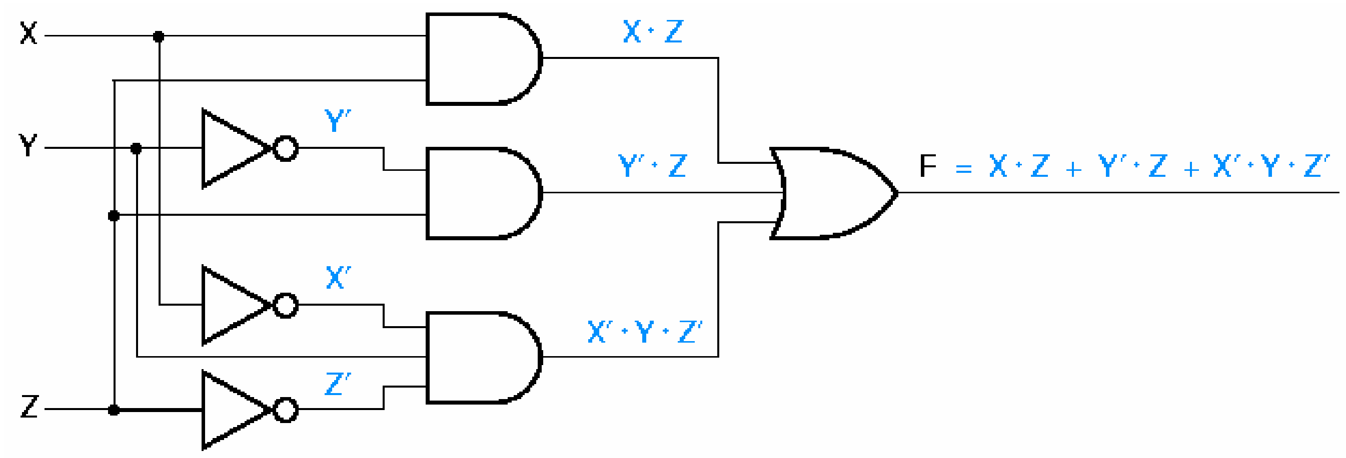
- Obtém-se F = ((X+Y') · Z) + (X'·Y·Z')

3. Conceitos sobre Sistemas Combinacionais

- Análise dum Circuito (4) -

- Após aplicar uma transformação algébrica obtém-se uma expressão e um circuito diferentes

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

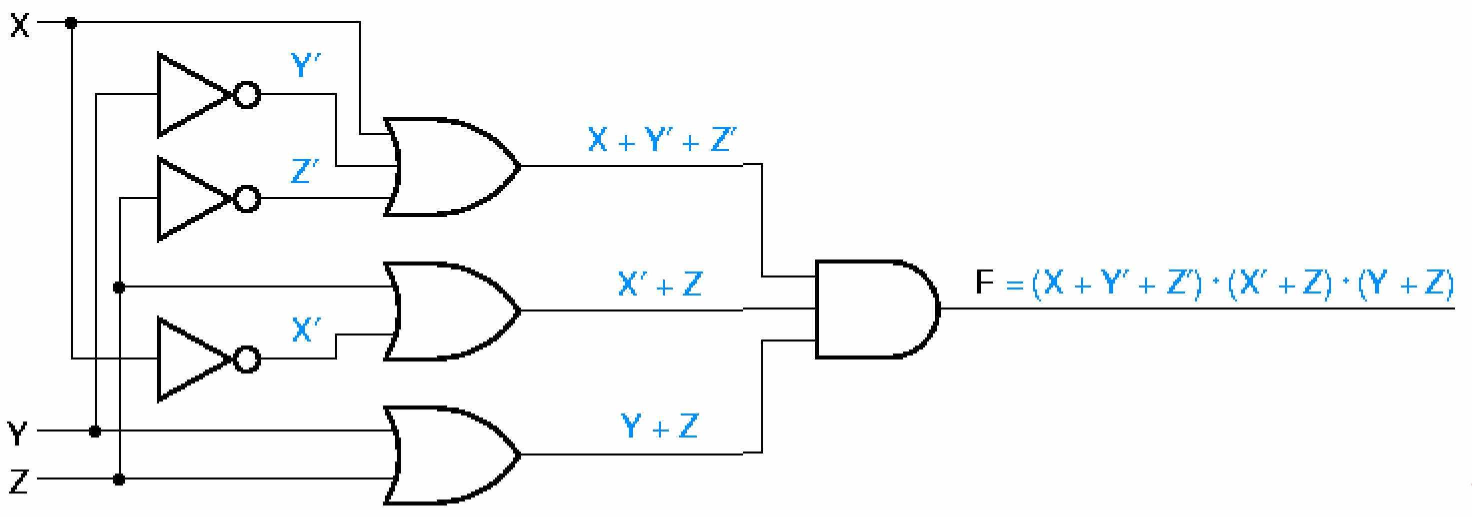


3. Conceitos sobre Sistemas Combinacionais

- Análise dum Circuito (5) -

- A partir da expressão de F original, também se pode derivar uma expressão do tipo POS (aplicando T8' e T5):

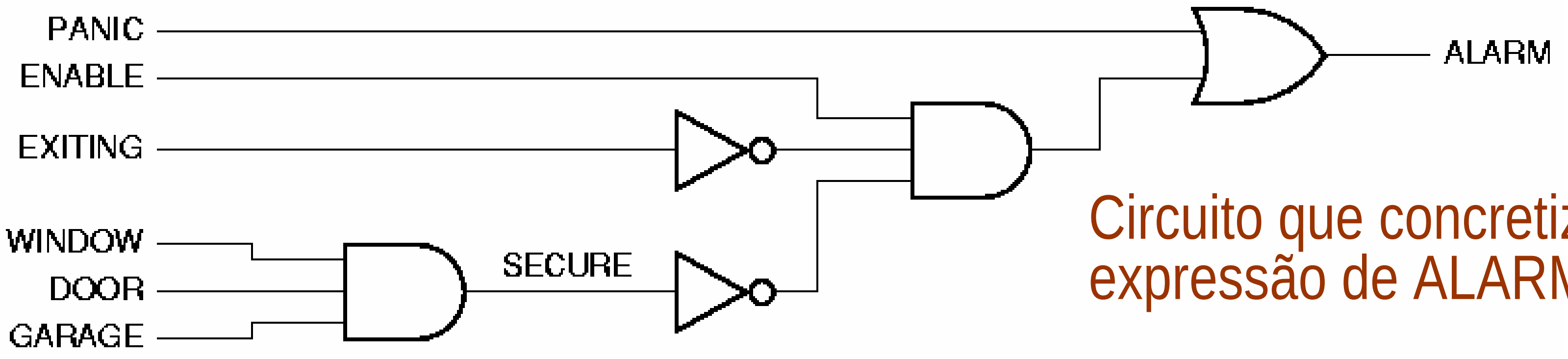
$$F = ((X+Y')\cdot Z) + (X'\cdot Y\cdot Z')$$
$$= (X+Y'+X')\cdot(X+Y'+Y)\cdot(X+Y'+Z')\cdot(Z+X')\cdot(Z+Y)\cdot(Z+Z') \qquad \text{[T8']}$$
$$= 1\cdot 1\cdot(X+Y'+Z')\cdot(Z+X')\cdot(Z+Y)\cdot 1 = \qquad \text{[T5]}$$
$$= (X+Y'+Z')\cdot(X'+Z)\cdot(Y+Z)$$



3. Conceitos sobre Sistemas Combinacionais

- Síntese dum Circuito (1) -

- O ponto de partida para projectar um circuito combinacional é normalmente a sua descrição em linguagem natural (por exemplo em português).
- <u>Exemplo</u>: construir um circuito de alarme."A saída ALARM é 1 se a entrada PANIC for 1 **ou** se a entrada ENABLE for 1, a entrada EXITING for 0 e a casa não estiver segura. A casa é segura se as entradas WINDOW, DOOR e GARAGE forem todas 1".
- ALARM = PANIC + ENABLE· EXITING'· SECURE'
  SECURE = WINDOW· DOOR· GARAGE

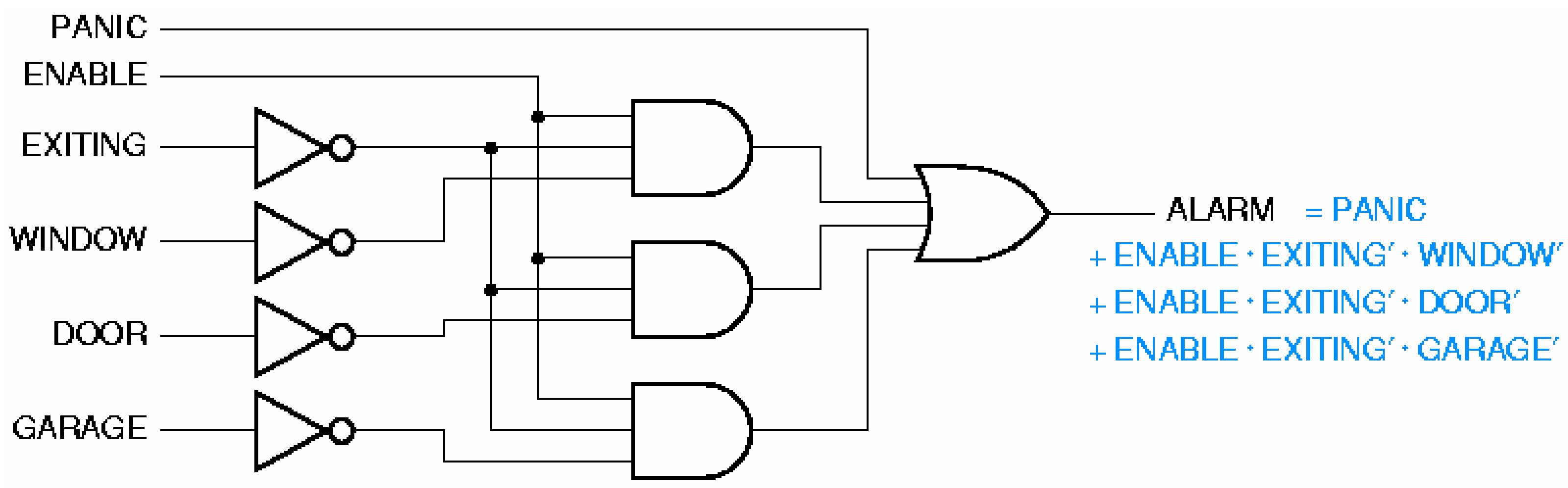


Circuito que concretiza a expressão de ALARM

3. Conceitos sobre Sistemas Combinacionais

- Síntese dum Circuito (2) -

- Concretizar a expressão de ALARM na forma SOP:
  - 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’



3. Conceitos sobre Sistemas Combinacionais

- Síntese dum Circuito (3) -

- Outras vezes, a descrição começa com uma listagem das combinações das entradas para as quais uma determinada saída deve estar activa ou inactiva (⇔ tabela de verdade dessa saída).

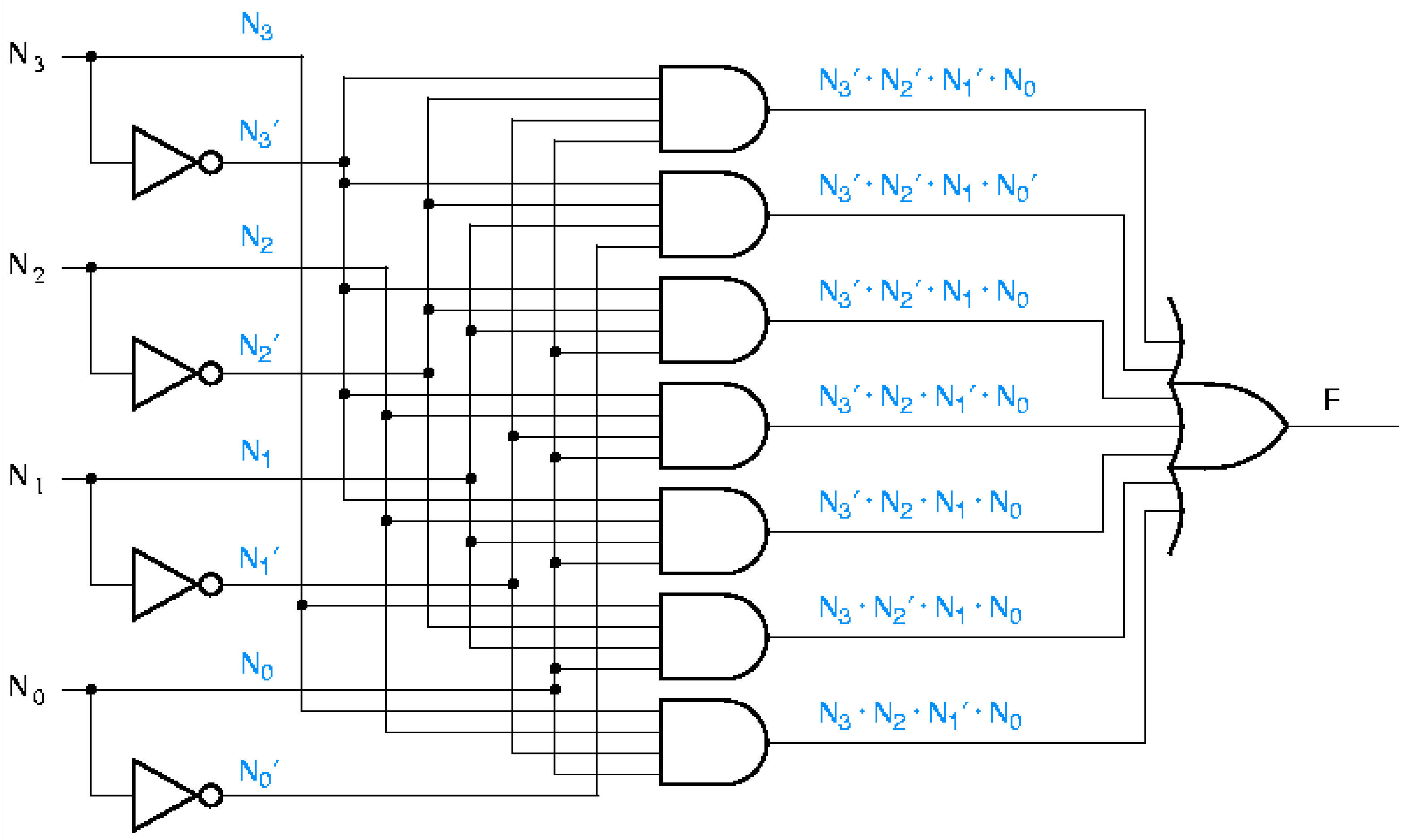
- Exemplo: Construir um circuito que detecte números primos de 4-bits. "Dada uma combinação de entrada $N=N_3N_2N_1N_0$ com 4-bits, o circuito gera 1 na saída quando N=1,2,3,5,7,11,13 e gera 0 nos outros casos."

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

3. Conceitos sobre Sistemas Combinacionais

- Síntese dum Circuito (4) -

- Concretização da expressão (SOP) da saída do detector



3. Conceitos sobre Sistemas Combinacionais

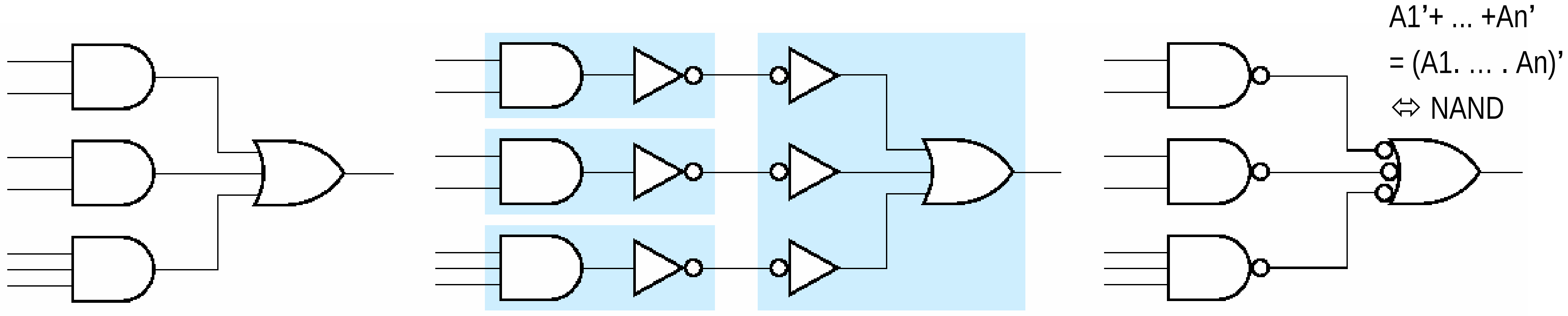
- Alterações a um Circuito (1) -

- Até aqui foram apresentados métodos para projectar circuitos que usam apenas portas AND, OR e NOT.
- Em certas situações, o projectista pode querer usar portas NAND ou NOR, dado que são mais rápidas que AND's e OR's na maioria das tecnologias.
- Contudo, a generalidade das pessoas não desenvolve proposições lógicas usando o NAND e o NOR como elementos de ligação.
- <u>Não se diz</u>: "Não gosto duma rapariga, se ela não for inteligente ou não for elegante **e** também se ela não for rica ou não for simpática"
  $G'=(I'+E')\cdot(R'+S')$
- <u>É mais frequente dizer-se</u>: "Gosto duma rapariga, se ela for inteligente e elegante **ou** se ela for rica e simpática"
  $G = (I\cdot E) + (R\cdot S)$
- Qualquer expressão lógica pode ser convertida numa expressão SOP equivalente e ser deste modo implementada com portas AND e OR.

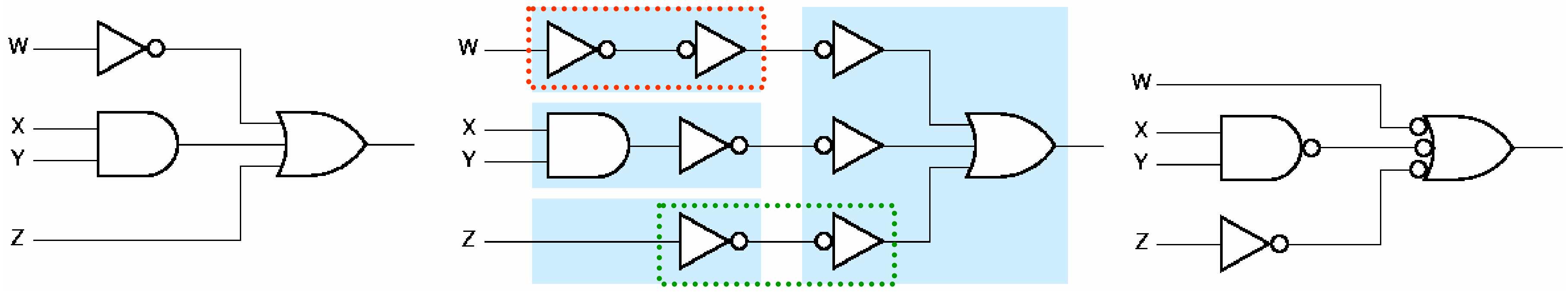
3. Conceitos sobre Sistemas Combinacionais

- Alterações a um Circuito (2) -

- Um circuito AND-OR com 2-níveis pode ser convertido num circuito NAND-NAND com 2-níveis, através duma simples substituição de portas.



- Se os termos (de produto) da expressão SOP incluirem apenas um literal, os inversores a aplicar a esse termo podem ser ou não necesssários.



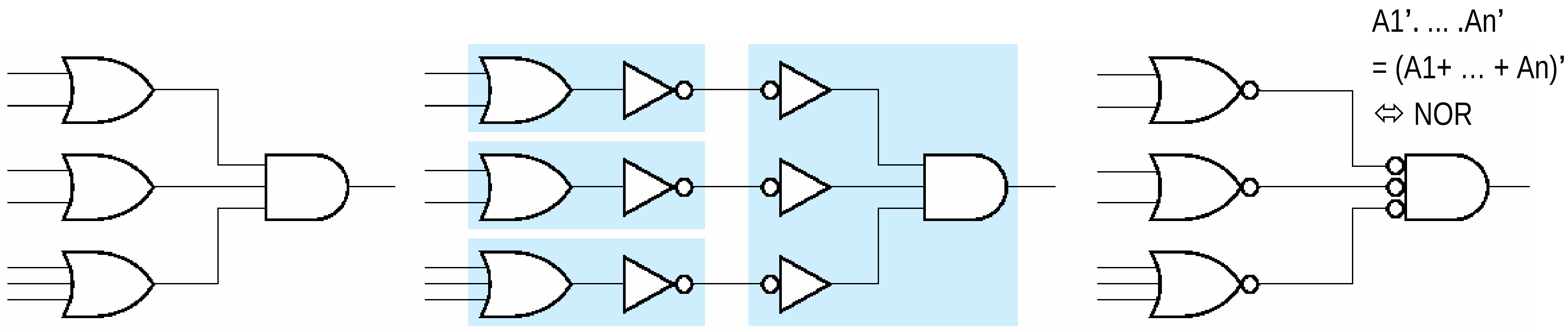
3. Conceitos sobre Sistemas Combinacionais

- Alterações a um Circuito (3) -

- Vimos que qualquer expressão SOP pode ser concretizada de duas formas: através dum circuito AND-OR ou dum circuito NAND-NAND.

- Aplicando o princípio da dualidade a esta regra, obtemos uma declaração que também é verdadeira:

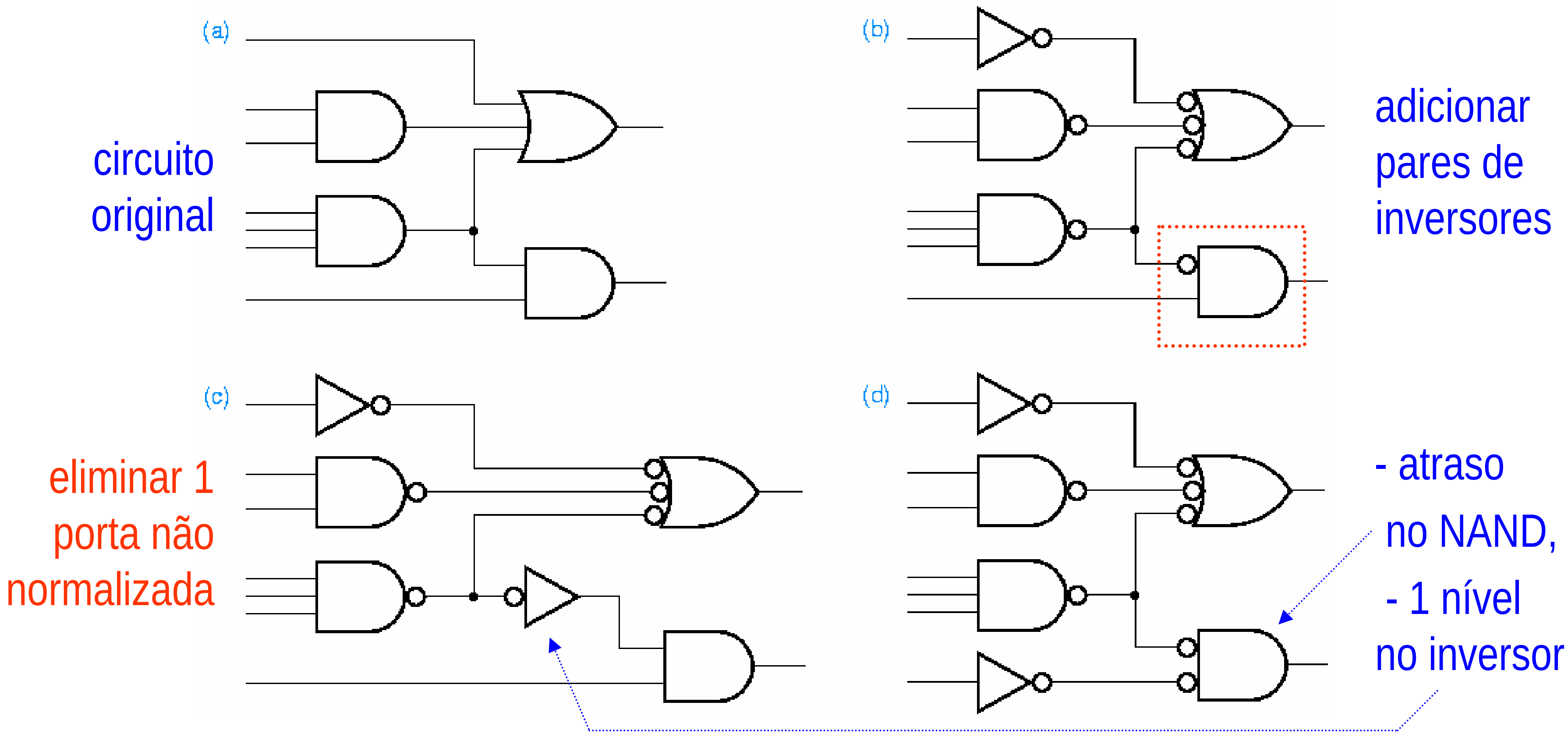
  Qualquer expressão POS pode ser concretizada de duas formas: através dum circuito OR-AND ou dum circuito NOR-NOR.



3. Conceitos sobre Sistemas Combinacionais

- Alterações a um Circuito (4) -

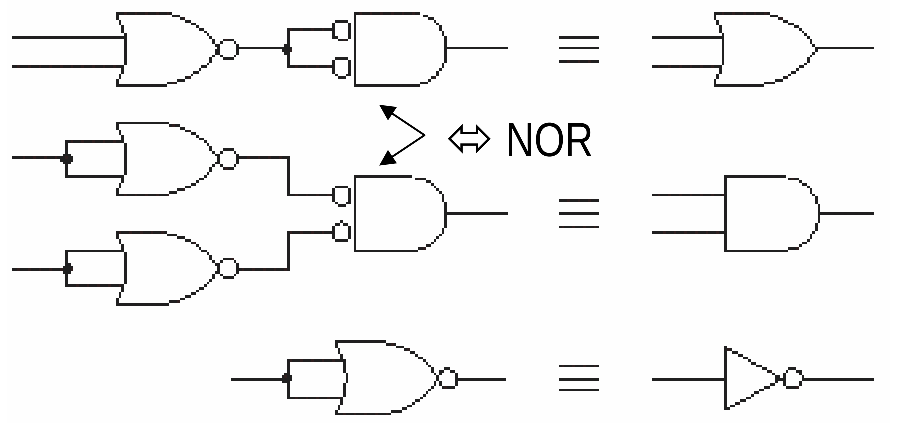
- Estas alterações (por ex., a conversão para a estrutura NAND-NAND) podem ser aplicadas a qualquer circuito lógico.



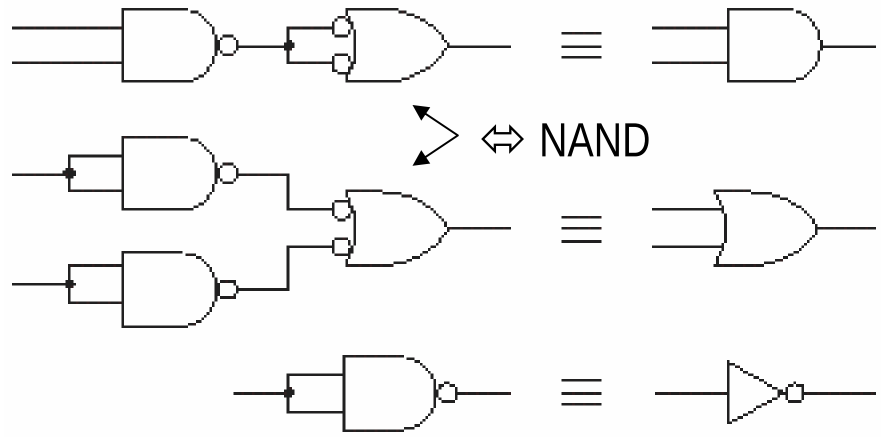
3. Conceitos sobre Sistemas Combinacionais

- Alterações a um Circuito (5) -

- Qualquer conjunto de tipos de porta lógica que permite concretizar qualquer função lógica é um **conjunto completo**.
- A porta AND de 2-entradas, a porta OR de 2-entradas e o inversor formam 1 conjunto completo



a porta NOR de 2-entradas é um conjunto completo



a porta NAND de 2-entradas é um conjunto completo

- Qualquer função lógica pode ser expressa numa soma de produtos de literais.
- As portas AND e OR, com qualquer número de entradas, podem ser construídas a partir de portas do mesmo tipo com 2-entradas.

3. Conceitos sobre Sistemas Combinacionais
- Minimização (1) -

- Normalmente não é económico concretizar uma função lógica directamente a partir da 1ª expressão que ocorre.
- As expressões canónicas (soma e produto) são especialmente consumidoras de recursos.
- A minimização de lógica emprega diversas técnicas para obter a implementação mais simples possível ao nível-da-porta para uma função.
- Contudo, o grau de simplificação depende da métrica usada.
- Três <u>métricas</u> que podem ser usadas:
  - número de literais
  - número de portas (lógicas)
  - número de níveis de portas em cascata

3. Conceitos sobre Sistemas Combinacionais

- Minimização (2) -

- O **número de literais** mede a quantidade de ligações necessária para implementar uma função.

- O **número de portas** mede a área (*espaço ocupado*) do circuito.

  Existe uma relação directa entre o número de portas dum projecto e o número de circuitos integrados necessário à sua implementação.

- O **número de níveis de portas** mede o número de portas entre as entradas e as saídas do circuito. Quanto maior é o número de níveis, maior é o atraso no circuito .

- Verifica-se que ao adequar um circuito para apresentar um atraso mínimo, raramente se consegue uma implementação com o menor nº de portas ou com as portas mais simples possíveis.

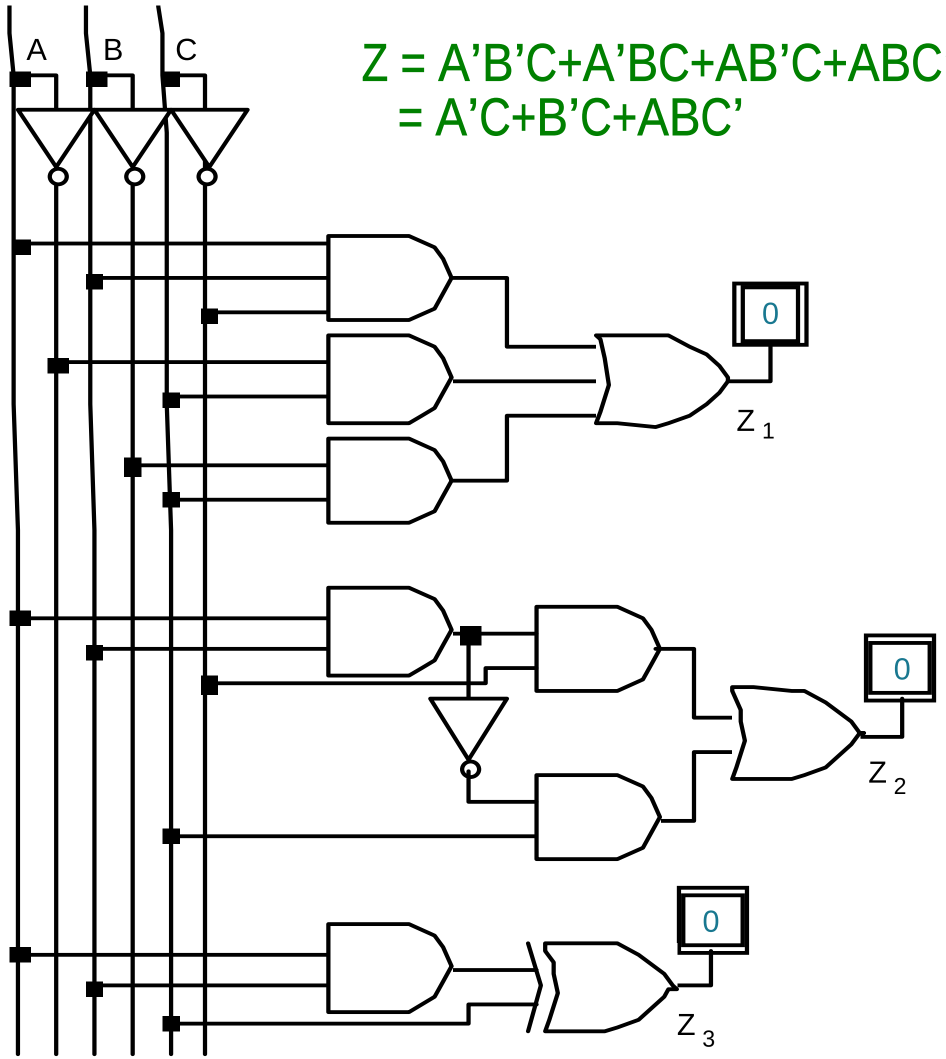
- Não é possível minimizar as três métricas ao mesmo tempo.

3. Conceitos sobre Sistemas Combinacionais

- Minimização (3) -

$Z = A'B'C+A'BC+AB'C+ABC'=A'(B'+B)C+(A'+A)B'C+ABC'$
$= A'C+B'C+ABC'$

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



Concretização a dois-níveis
(os inversores não contam)
- Número de literais maior

Concretização multi-nível
+ Portas com menos entradas
- Número de níveis maior

$Z = ABC'+A'C+B'C$
$= (AB)C'+(A'+B')C$
$= (AB)C'+(AB)'.C$

Porta mais complexa : XOR
+ Menos portas
- Atraso maior

$Z = (AB)C'+(AB)'.C=(AB)\ \text{xor}\ C$

3. Conceitos sobre Sistemas Combinacionais

- Minimização (4) -

- As técnicas de minimização reduzem o número e o tamanho das portas necessárias para construir um circuito, diminuindo assim o custo do sistema.

- Os métodos de minimização reduzem o custo dum circuito AND-OR ou OR-AND a 2-níveis através de:

  - minimizar o número de portas no primeiro nível;
  - minimizar o número de entradas em cada porta do primeiro nível;
  - minimizar o número de entradas em cada porta do segundo nível.

3. Conceitos sobre Sistemas Combinacionais

- Minimização (5) -

- Os métodos de minimização não consideram o custo dos inversores à entrada.
- Consideram que todas as variáveis de entrada, e seus complementares, estão disponíveis (adequado para implementações com PLD's).
- Também assumem que a função a minimizar está representada por uma tabela de verdade ou por uma lista de mintermos ou maxtermos.
- A minimização baseia-se nos teoremas T10 e T10':
  $\text{produto} \cdot Y + \text{produto} \cdot Y' = \text{produto}$
  $(\text{soma} + Z) \cdot (\text{soma} + Z') = \text{soma}$
- Se dois termos diferem apenas numa variável, podem ser substituídos por um único termo com menos uma variável.
- Poupa-se uma porta e a outra porta possui menos uma entrada.

3. Conceitos sobre Sistemas Combinacionais

- Minimização (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

O valor de B varia nas linhas do on-set

B é eliminado , *A mantém-se*

O valor de A não varia nas linhas do on-set

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

G = A' B' + A B' = (A' + A) B' = B'

O valor de B não varia nas linhas do on-set

A é eliminado , *B mantém-se*

O valor de A varia nas linhas do on-set

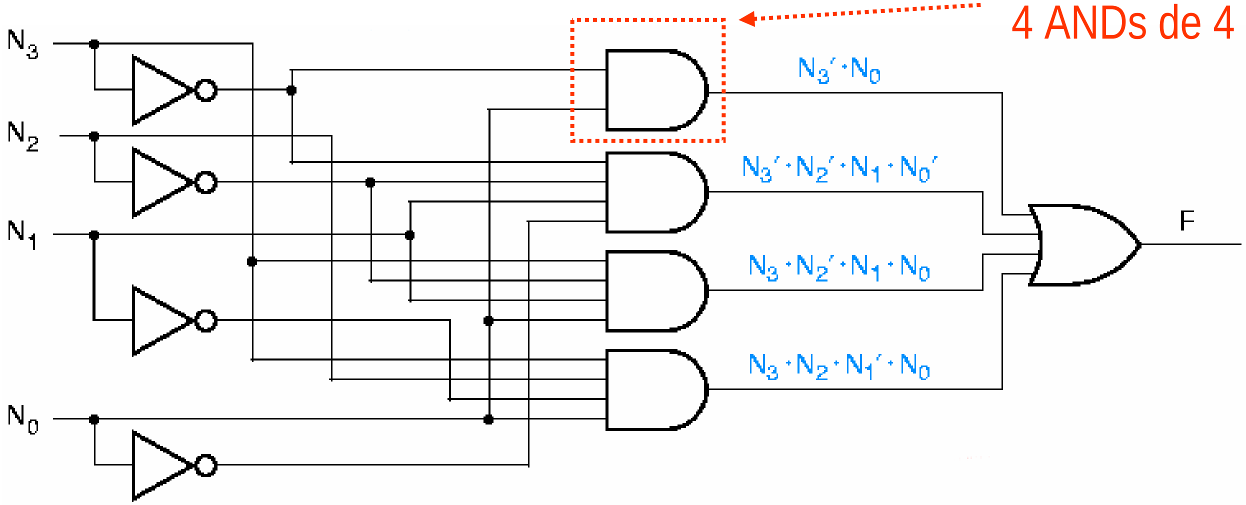
Essência da simplificação:
Encontrar pares de elementos do ON-set em que apenas uma variável muda de valor. A variável que muda de valor pode ser eliminada.

3. Conceitos sobre Sistemas Combinacionais

- Minimização (7) -

- Vamos aplicar esta técnica à expressão do detector de números primos.
- $F = \sum_{N_3,N_2,N_1,N_0} m(1,2,3,5,7,11,13) =$
  $N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0 + N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0 + \ldots =$
  $(N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0) + (N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0) + \ldots =$
  $(N_3' \cdot N_2' \cdot N_0) + (N_3' \cdot N_2 \cdot N_0) + \ldots = N_3' \cdot N_0 + \ldots$

resultado de simplificar
4 ANDs de 4 entradas

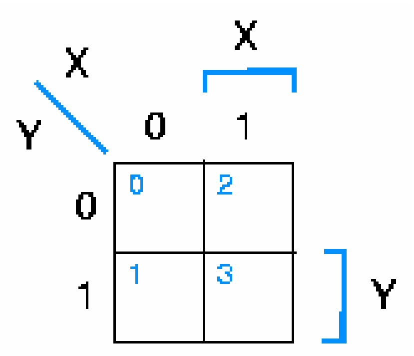


3. Conceitos sobre Sistemas Combinacionais

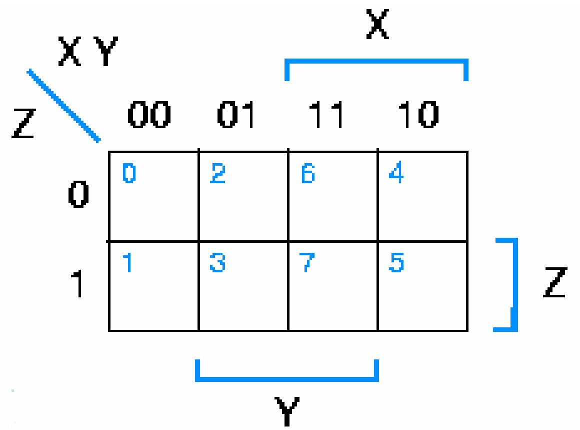
- Mapas de Karnaugh (1) -

- Não é fácil encontrar o par de termos que participa em cada simplificação.
- Um **mapa de Karnaugh** é uma representação gráfica para a tabela de verdade duma função lógica.
- O mapa para uma função lógica de n-entradas é um array com 2^n células, uma por cada mintermo.

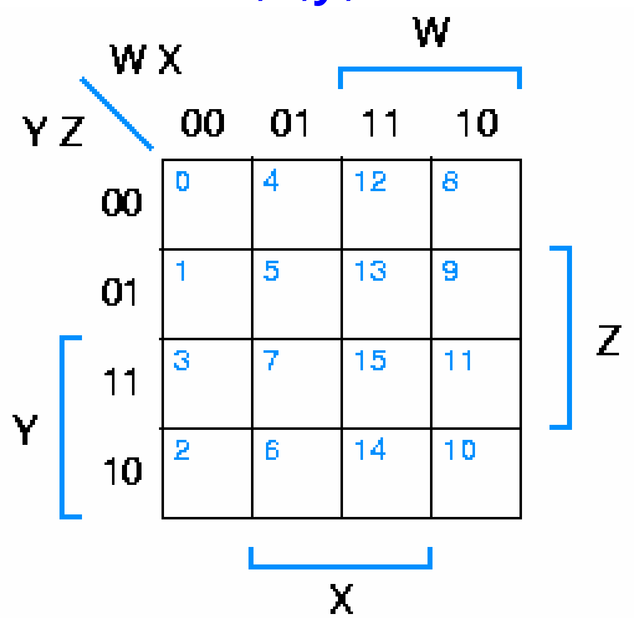
F x,y



G x,y,z



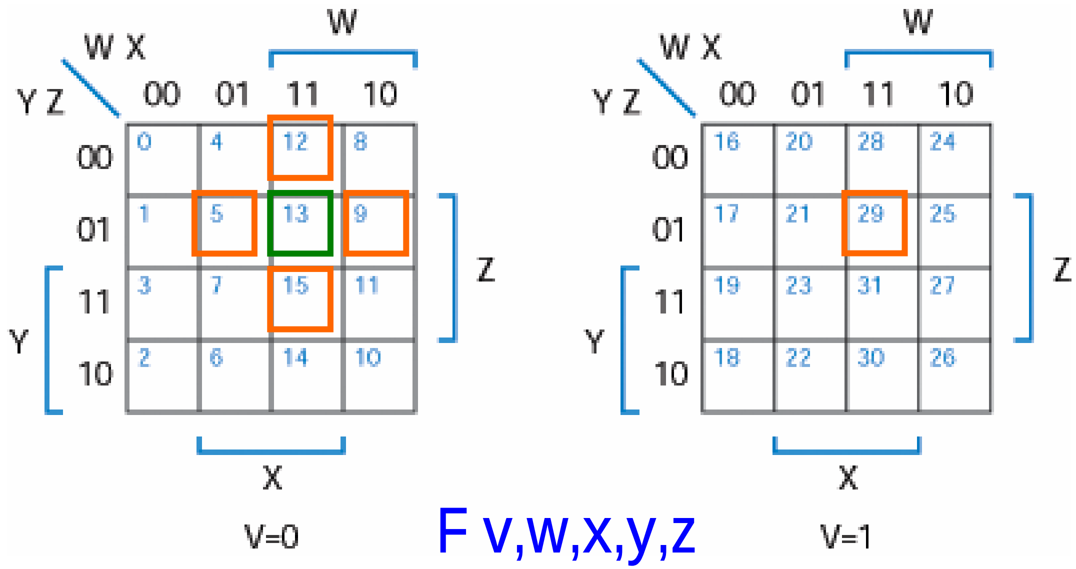
H w,x,y,z



3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (2) -

- Os mapas de Karnaugh que se utilizam para representar funções com 5 e 6 variáveis não são tão adequados como os mapas de 2, 3 e 4 variáveis, dado a adjacência ser mais difícil de visualizar.
- Num mapa de **5 variáveis**, é necessário recorrer a 2 mapas de 4 variáveis colocados um ao lado do outro.
- Nesta representação, considera-se que um mapa é colocado por cima do outro, de modo a originar um objecto 3-dimensional.
- Cada célula é adjacente de 5 células (4 no mesmo mapa e 1 no outro).



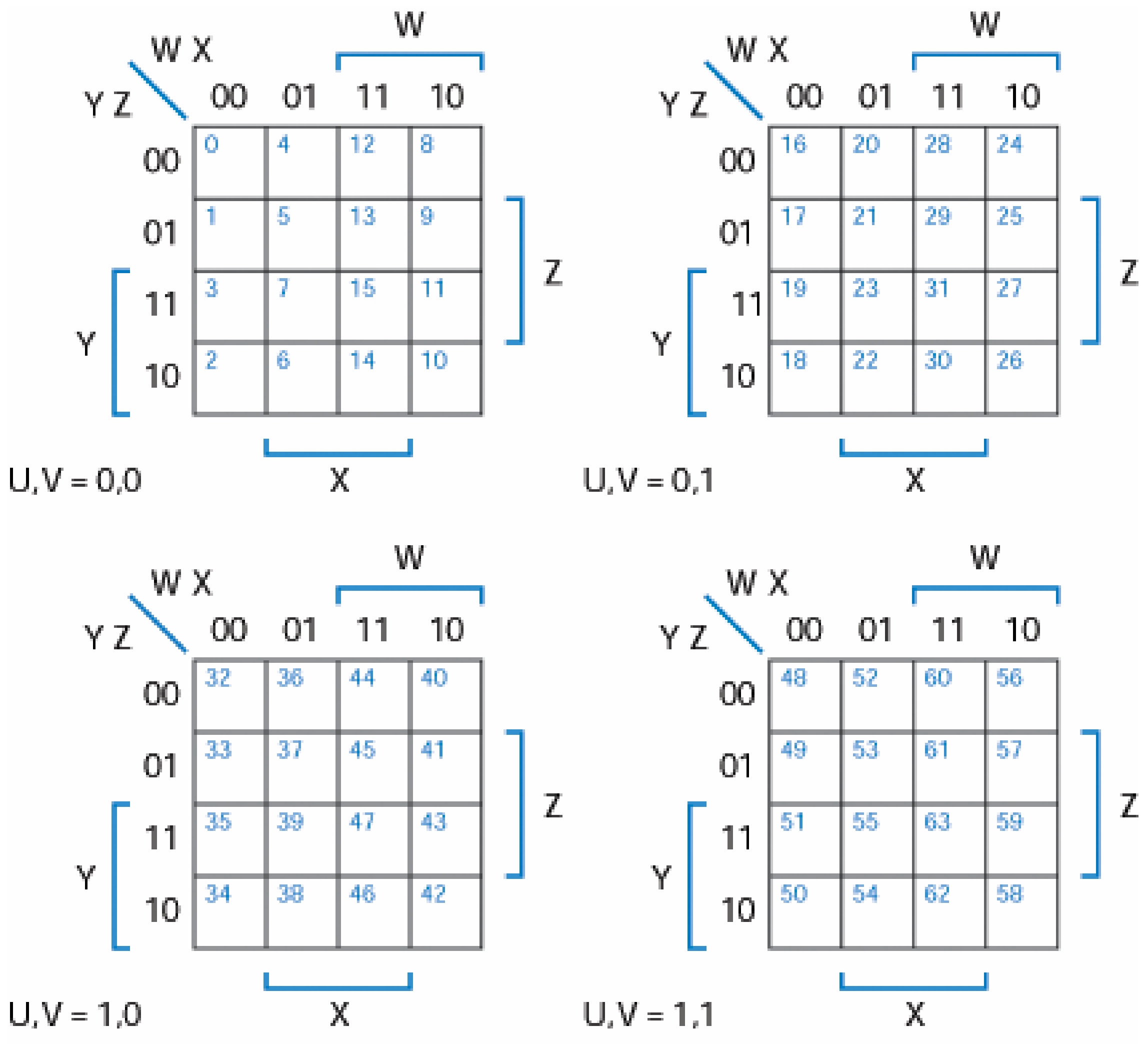
F v,w,x,y,z

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (3) -

Um mapa de Karnaugh com 6 variáveis:

U,V,W,X,Y,Z

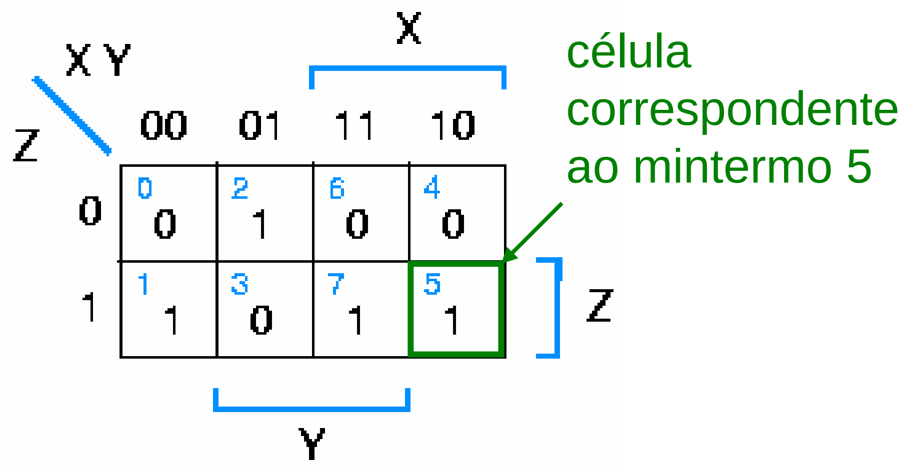


3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (4) -

- Para representar uma função lógica num mapa de Karnaugh, copiam-se os 1's e 0's da tabela de verdade para as células correspondentes do mapa.
- Cada célula do mapa corresponde a um mintermo da função.

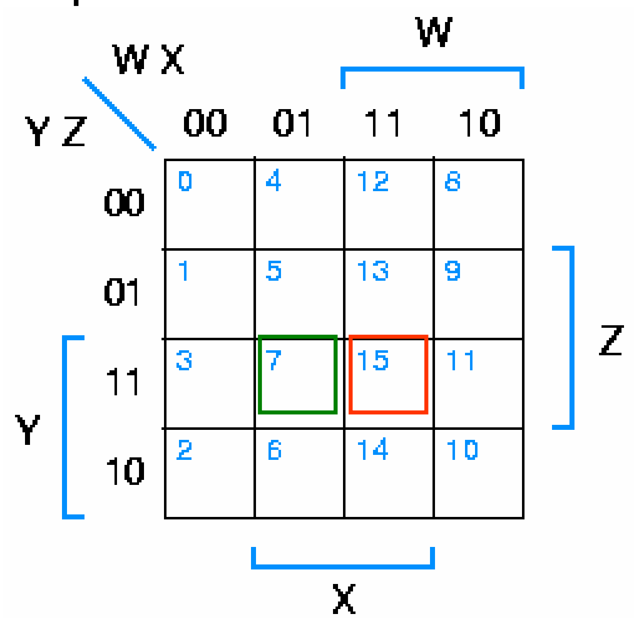
X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



- Na prática, apenas se copiam os 1's ou os 0's (não ambos) para as células, dependendo do tipo de expressão que se pretende obter (SOP ou POS).

3. Conceitos sobre Sistemas Combinacionais
- Mapas de Karnaugh (5) -

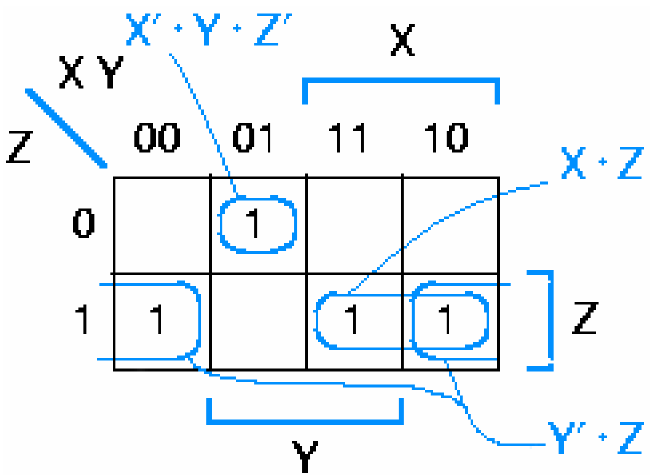
- Porque se usa uma ordenação estranha das linhas/colunas?
  Porque assim, cada célula corresponde a uma combinação das entradas que difere apenas numa variável da combinação associada a cada uma das células vizinhas imediatamente adjacentes.
- As células 7 e 15 no mapa de 4 variáveis diferem apenas no valor de W.
- Nos mapas de 3 e 4 variáveis, as células no canto esquerdo/direito ou superior/inferior também são vizinhas.
- As células 8 e 10 no mapa de 4 variáveis diferem apenas no valor de Y.
- Uma vez que os pares de células-a-1 adjacentes correspondem a mintermos que diferem apenas numa variável, cada par pode ser combinado num único termo de produto, usando o teorema T10:
  produto·Y + produto·Y' = produto



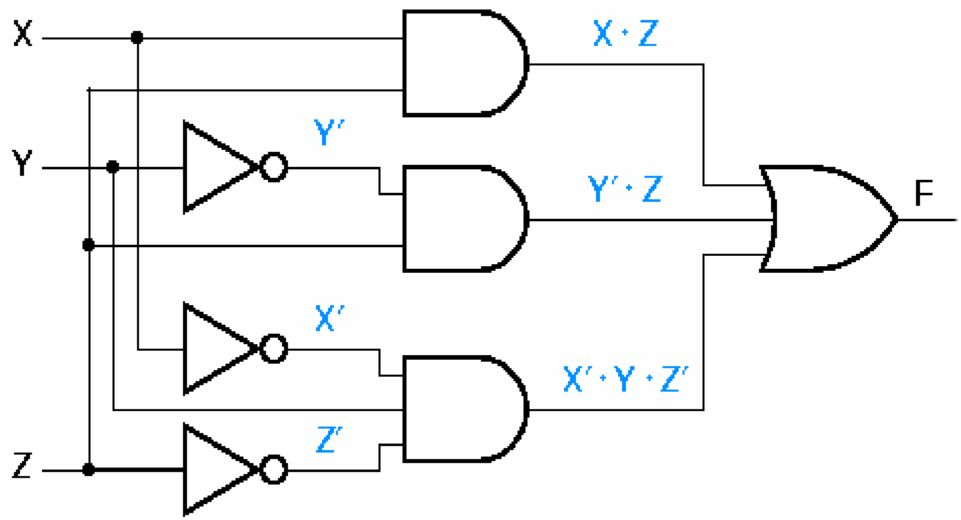
3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (6) -

- Simplificar F com mapas Karnaugh



- Combinando as células 5 e 7:
  F = ... + X·Y'·Z + X·Y·Z
  = ... + X·Z
- Combinando as células 1 e 5:
  F = X'·Y'·Z + X·Y'·Z + ...
  = Y'·Z + ...
- O mintermo 5 é incluído duas vezes → não há problema pois X+X=X.
- F = X·Z + Y'·Z + X'·Y·Z'



concretização AND-OR de F simplificada

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (7) -

- O procedimento utilizado para combinar células, pode ser estendido para permitir combinar **mais do que 2 células-a-1** num único termo.

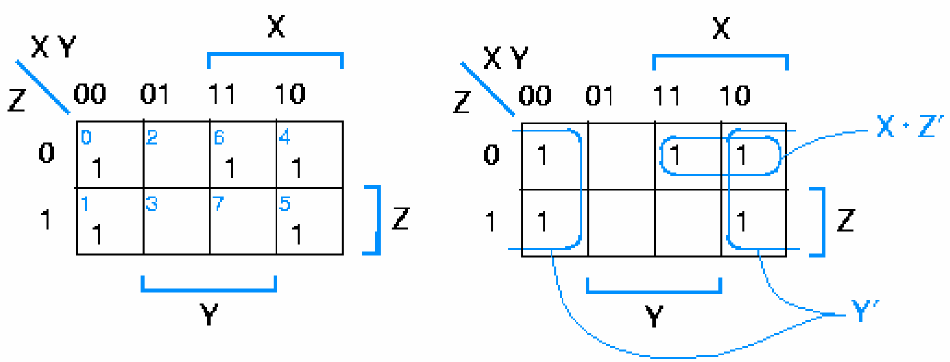
- $F = \sum_{X,Y,Z,} m(0,1,4,5,6) = X'\cdot Y'\cdot Z' + X'\cdot Y'\cdot Z + X\cdot Y'\cdot Z' + X\cdot Y'\cdot Z + X\cdot Y\cdot Z'$

  $= [Y'\cdot(X'\cdot Z') + Y'\cdot(X'\cdot Z) + Y'\cdot(X\cdot Z') + Y'\cdot(X\cdot Z)] + X\cdot Y\cdot Z'$

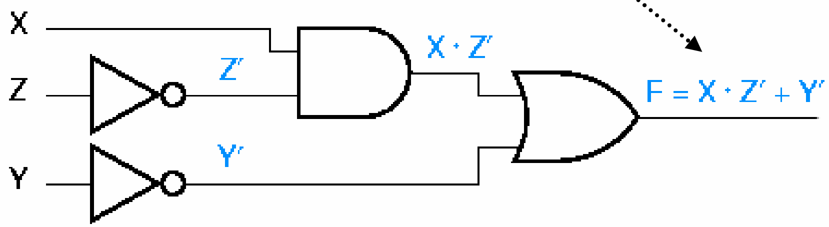
  $= Y'\cdot[(X'\cdot Z' + X\cdot Z') + (X'\cdot Z + X\cdot Z)] + X\cdot Y\cdot Z' = Y'\cdot[(X' + X).Z' + (X' + X)\cdot Z] + X\cdot Y\cdot Z'$

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

diferentes
porquê?



simplificação com mapa de Karnaugh



concretização AND-OR de F simplificada
com mapa de Karnaugh

3. Conceitos sobre Sistemas Combinacionais

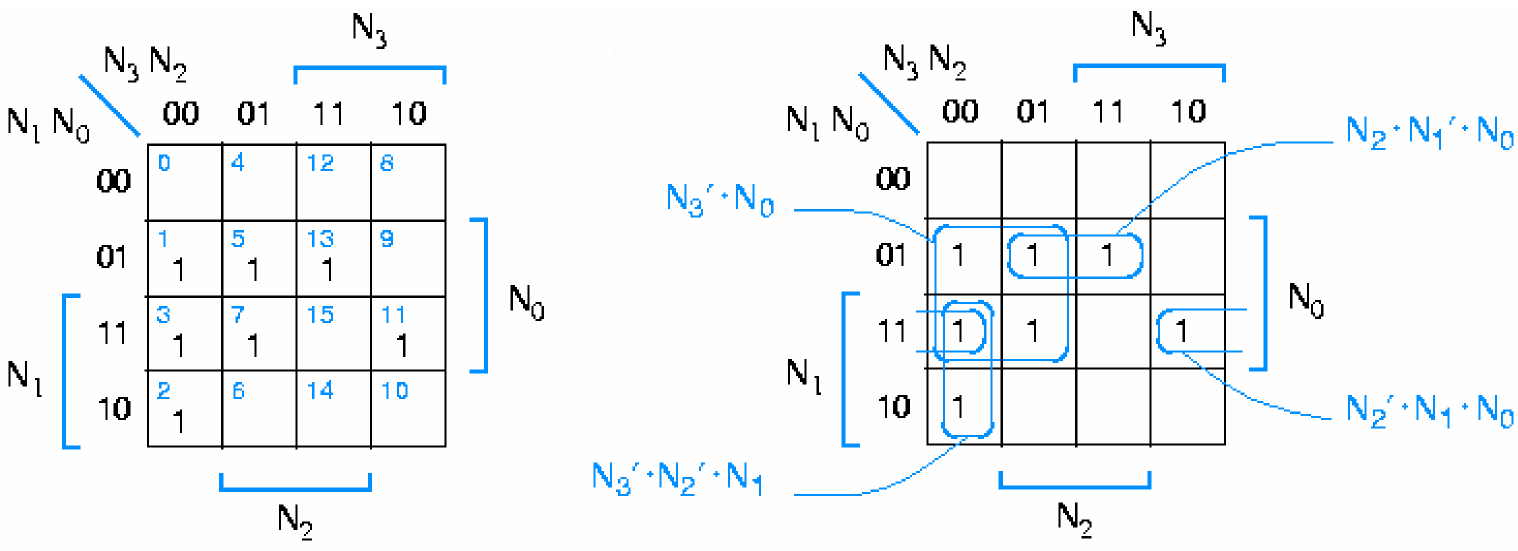
- Mapas de Karnaugh (8) -

- Generalizando, pode combinar-se 2^i **células-a-1** para originar um termo de produto com ***n-i*** literais (em que **n** = número de variáveis).
- Regra para combinar as células-a-1:
  - Um conjunto de 2^i células-a-1 pode ser combinado se existirem *i* variáveis que assumem todas as possíveis combinações (2^i) dentro desse conjunto, enquanto as restantes *n-i* variáveis mantêm o mesmo valor em todo o conjunto.
  - O termo de produto resultante possui *n-i* literais, em que cada literal é: a variável complementada (se ela aparecer como 0 em todas as células-a-1) ou a variável não complementada (se ela aparecer como 1).
- Graficamente: podemos envolver conjuntos rectangulares de 2^n células-a-1 com um "rectângulo".

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (9) -

- A partir de cada rectângulo que envolve células-a-1, obtém-se o correspondente termo de produto:
  - Se o rectângulo cobre apenas zonas do mapa em que uma dada variável é 0 (1), então essa variável surge complementada (não complementada) no termo de produto.
  - Se o rectângulo cobre zonas do mapa em que uma dada variável é 0 e 1, então essa variável não aparece no termo de produto.



$F = \Sigma_{N3,N2,N1,N0}(1,2,3,5,7,11,13)$

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

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (10) -

- Uma **soma mínima** para a função lógica F é uma expressão do tipo soma de produtos (SOP) para F tal que nenhuma outra expressão SOP para F possui menos termos de produto, e qualquer expressão SOP com o mesmo número de termos de produto possui pelo menos tantos literais como ela.
- A soma mínima possui o menor nº de termos de produto possível (nº de portas no 1º nível e nº de entradas na porta do 2º nível) e o menor nº de literais possível (nº de entradas nas portas do 1º nível).
- Uma função lógica **P implica** a função lógica **F** ($P \Rightarrow F$) se para cada combinação de entradas em que P=1, então também F=1. Ou seja, F <u>inclui</u> ou <u>cobre</u> P.
- Um **implicante maior** duma função lógica F é um termo de produto normal P que implica F, de tal modo que se qualquer variável for eliminada de P, então o termo de produto resultante já não implica F.

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (11) -

- Em termos dum mapa de Karnaugh, um <u>implicante maior</u> de F é um conjunto de células-a-1 envolvido por um rectângulo, de tal modo que se tentarmos aumentá-lo (de modo a cobrir o dobro das células), ele vai cobrir um ou mais 0's.
- **Teorema do implicante maior**: uma soma mínima é uma soma **de** implicantes maiores.
- Para encontrar uma <u>soma mínima</u>, **não é** necessário ter em conta qualquer termo de produto que não seja um implicante maior.
- A soma de todos os implicantes maiores duma função é designada por **soma completa**.
- A <u>soma completa</u> **não é** necessariamente uma soma mínima.

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (12) -

Algoritmo: obter a expressão SOP mínima com um mapa de Karnaugh

Passo 1: Marcar os implicantes maiores

Escolher um "1" do ON-set ainda não visitado.

Encontrar os grupos de 1's (e X's) adjacentes desse elemento e que possuem a maior dimensão possível. A dimensão tem que ser 2^i. Não é preciso repetir grupos já considerados.

Não esquecer a adjacência entre: a linha superior e inferior, a coluna esquerda e direita e os cantos.

Repetir o Passo 1 para cada "1" do ON-set, de modo a encontrar todo os implicantes maiores.

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (13) -

Passo 2: Marcar os implicantes essenciais

Visitar cada “1” do ON-set. Se o “1” for coberto por um único implicante maior, então este implicante é essencial e aparece na expressão final. Os restantes 1’s cobertos pelo implicante não precisam ser revisitados.

Repetir o Passo 2 até que todos os implicantes essenciais tenham sido encontrados.

Passo 3: Cobertura adicional

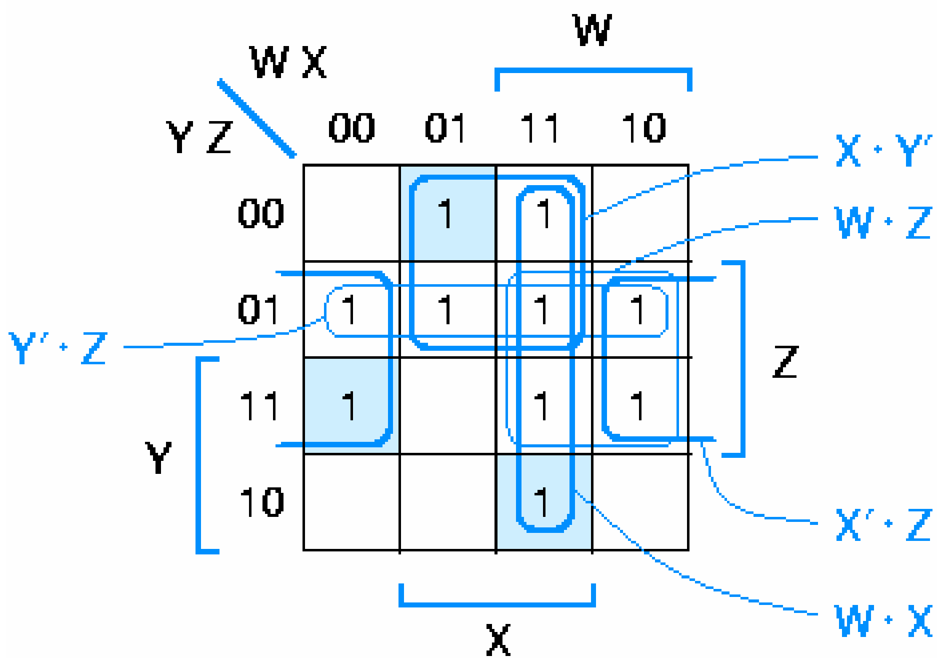
Se existirem 1’s não cobertos pelos implicantes essenciais, seleccionar o menor número de implicantes maiores que cubra todos esses 1’s. Tentar várias alternativas de cobertura.

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (14) -

- $F = \sum_{W,X,Y,Z,} m(1,3,4,5,9,11,12,13,14,15)$
- A função possui 5 implicantes maiores.
- A soma mínima inclui apenas 3 implicantes maiores:

  $F = X \cdot Y' + X' \cdot Z + W \cdot X$
- Como se decide quais os implicantes maiores a incluir na soma mínima duma função?



3. Conceitos sobre Sistemas Combinacionais

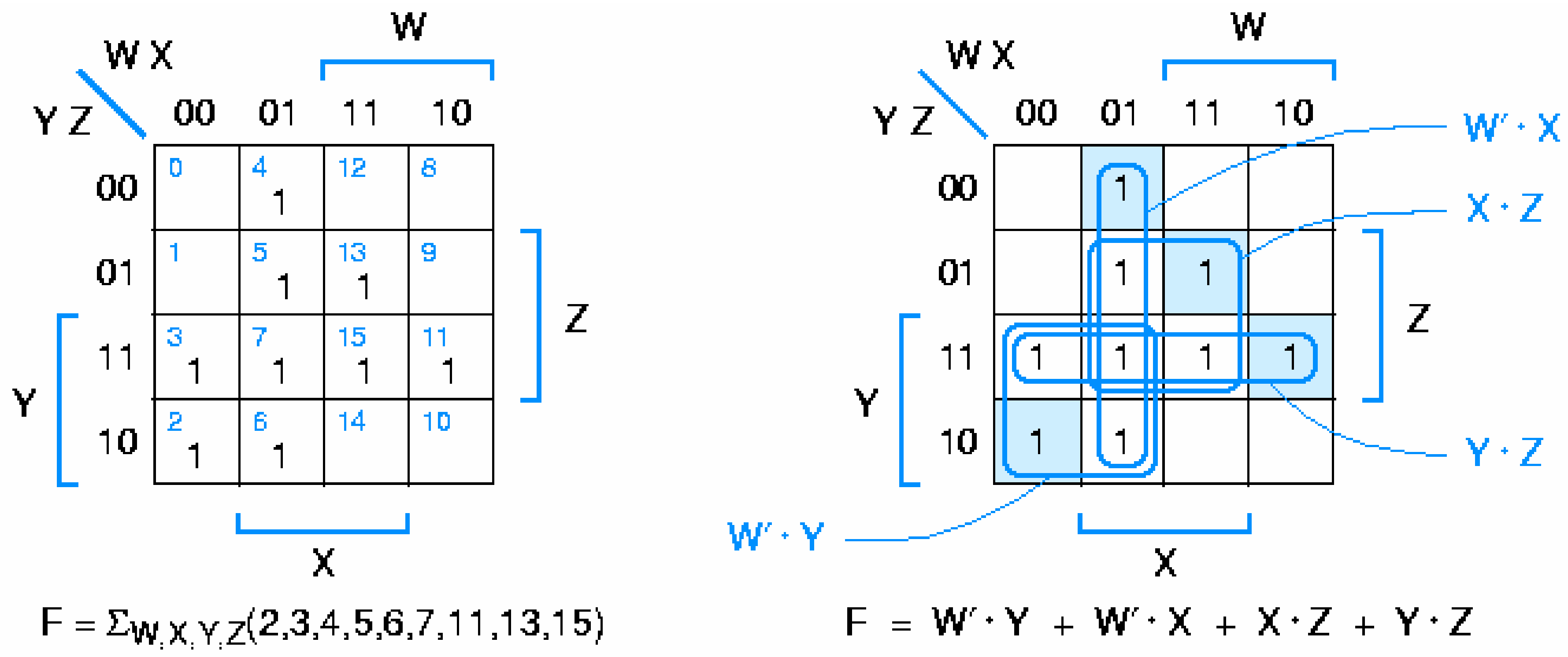
- Mapas de Karnaugh (15) -

- Uma **célula-a-1 distinguida** duma função lógica é uma combinação de entradas que é coberta por um único implicante maior.
- Um **implicante essencial** duma função lógica é um implicante maior que cobre uma ou mais células-a-1distinguidas.
- Os implicantes essenciais <u>têm</u> que aparecer em qualquer soma mínima.
- O 1º passo na selecção dos implicantes consiste em identificar as células-a-1 distinguidas e incluir os correspondentes implicantes essenciais na soma.
- Depois, se houver células-a-1 não cobertas pelos implicantes essenciais, falta encontrar a melhor forma de as cobrir.

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (16) -

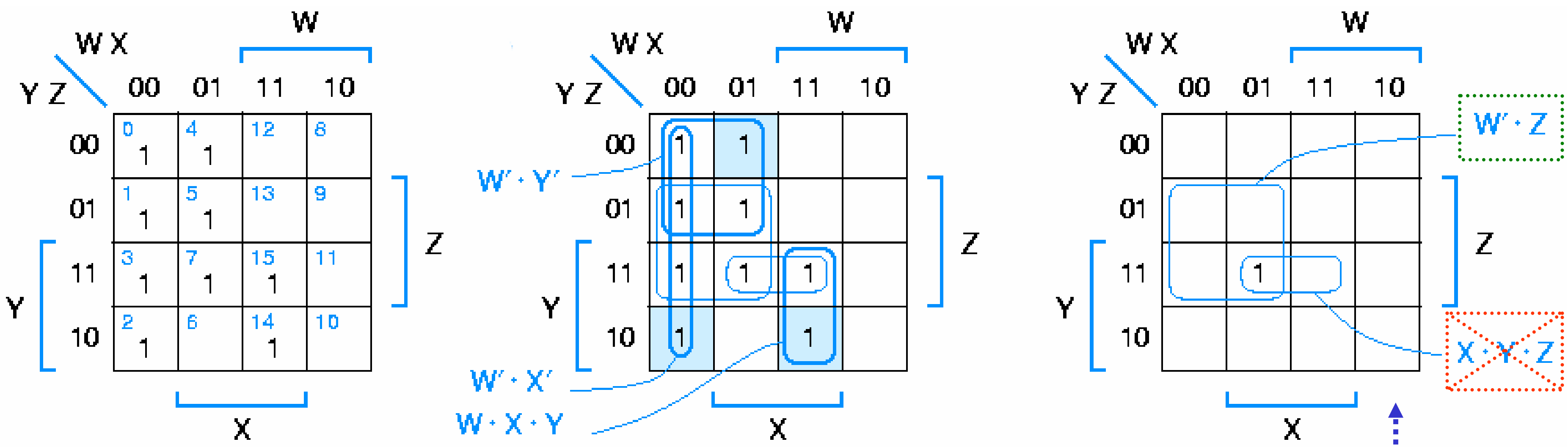
- Exemplo em que todos os implicantes maiores são essenciais



3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (17) -

- Exemplo em que nem todos os implicantes maiores são essenciais



$F = \Sigma_{W,X,Y,Z}(0,1,2,3,4,5,7,14,15)$

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

mapa que se obtém ao retirar os implicantes essenciais e os 1's que eles cobrem

simplifica mais que

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (18) -

- Aplicando o princípio da dualidade, pode minimizar-se expressões do tipo produto de somas (POS) se se funcionar com os 0's do mapa de Karnaugh.

  Cada 0 do mapa corresponde a um maxtermo.

- Uma forma mais fácil de encontrar o produto mínimo de F consiste em obter a soma mínima de **F'**.

- Obter F**'** é simples: os 1's de F**'** são os 0's de F.

- Após obter a soma mínima para F**'** *(F'sop)*, complementa-se o resultado obtido aplicando o teorema de DeMorgan generalizado [T14] , de modo a obter o produto mínimo para F *(Fpos)*.

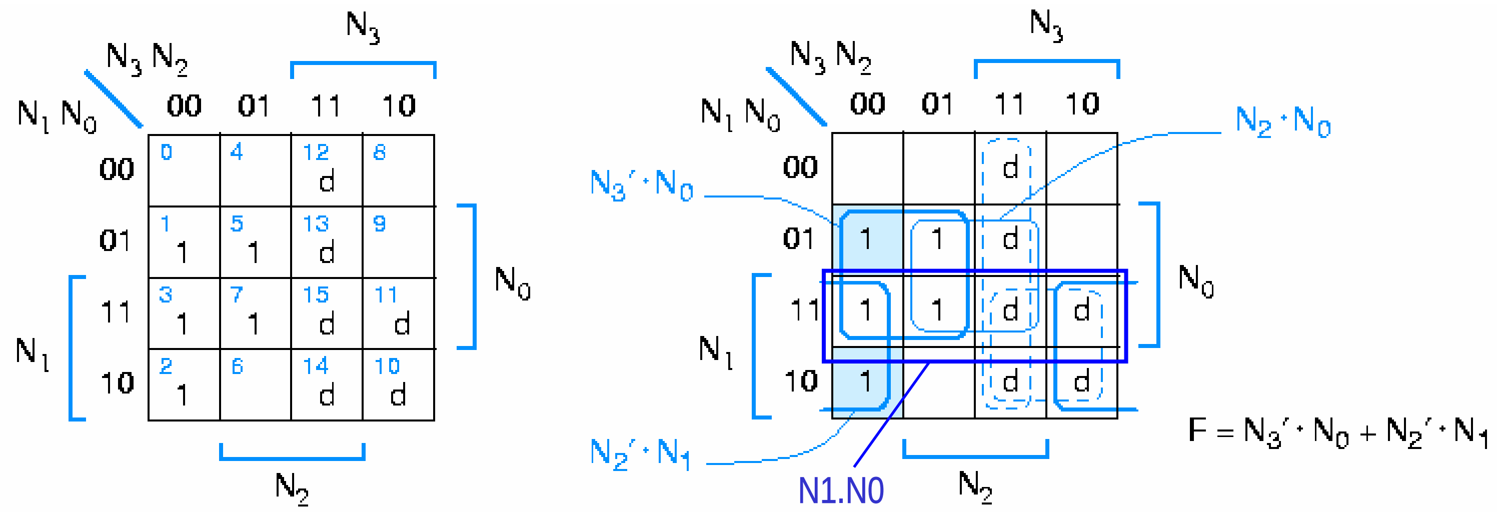
- Exemplo: $F' = X \cdot Y' + X' \cdot Z + W \cdot X$

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

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (19) -

- Por vezes, não interessa qual é o valor da saída da função para certas combinações das entradas. A estas saídas chama-se *don't cares*.
- Exemplo: um detector de números primos em que a entrada *N* com 4-bits é sempre um dígito BCD, ou seja, os mintermos 10-15 nunca ocorrem.
- $F = \sum_{N_3,N_2,N_1,N_0} m\,(1,2,3,5,7) + \sum d\,(10,11,12,13,14,15)$



3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (20) -

- Ao introduzir *don't cares* (X's), o procedimento usado para envolver os conjuntos de 1's com rectângulos é alterado do seguinte modo:
  - Permite-se que os X's sejam incluídos nos conjuntos de 1's a envolver com rectângulos, para formar conjuntos tão grandes quanto possível. Desta forma reduz-se o número de variáveis nos implicantes maiores que lhe correspondem.
  - Não se considera qualquer conjunto que contenha apenas X's. Incluir na função os termos de produto formados a partir de X's aumentaria desnecessariamente o seu custo (⇔ a função ficaria menos minimizada).
- O resto do procedimento mantém-se válido. Por exemplo:
  - Visitam-se apenas as células-a-1distinguidas e não as células-a-X distinguidas.
  - Inclui-se na função simplificada apenas (i) os implicantes essenciais que correspondem a esses 1's e (ii) outros implicantes maiores que sejam necessários para cobrir todos os 1's do mapa.

3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (21) -

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

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

Mapa de Karnaugh inicial

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

Impl. maiores envolvendo m_4=A' B C' D'

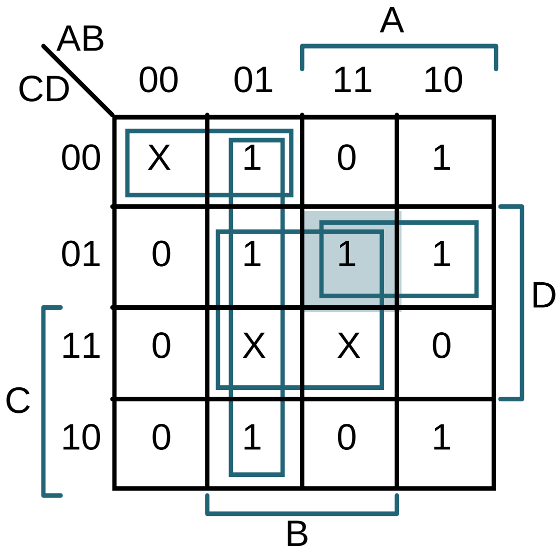
CD \ AB	00	01	11	10
00	X	1	0	1
01	0	1	1	1
11	0	X	X	0
10	0	1	0	1

Impl. maiores envolvendo m4 e m_5=A' B C' D

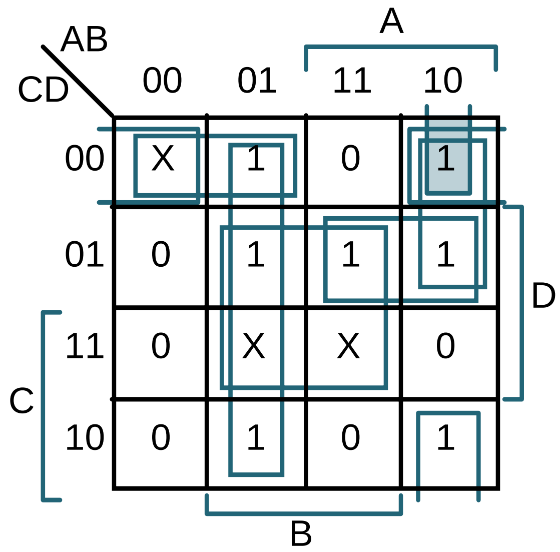
3. Conceitos sobre Sistemas Combinacionais

- Mapas de Karnaugh (22) -

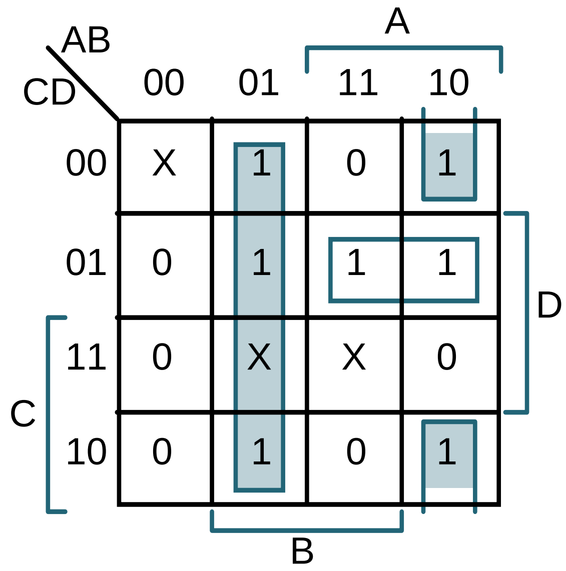
Exemplo (continuação)



Impl. maiores envolvendo m4, m5, m6 e m_{13}=A B C' D'



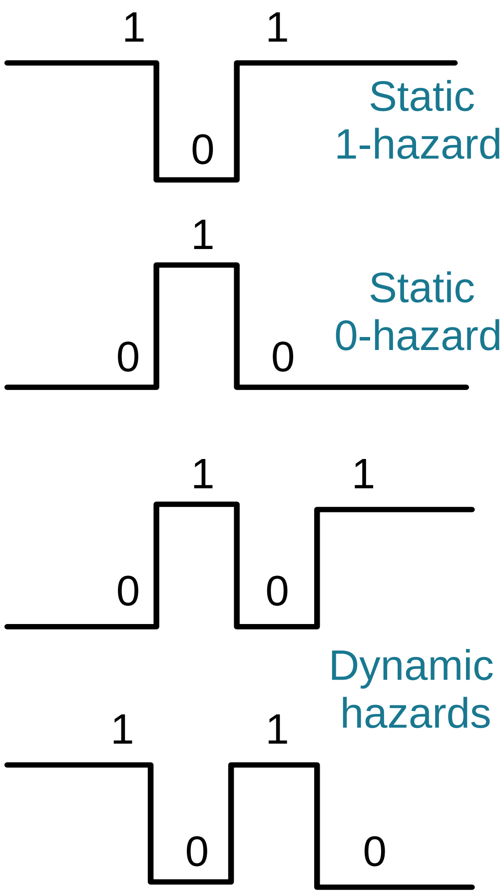
Impl. maiores envolvendo m4-6, m13 e m_8=A B' C' D'



Implicantes essenciais e cobertura mínima para os 1's restantes

3. Conceitos sobre Sistemas Combinacionais
- Hazards (1) -

- Devido aos atrasos nos componentes electrónicos, um circuito pode originar um *glitch*.
- Um *glitch* é uma variação de curta duração no valor duma saída, quando não se espera nenhuma variação.
- Ocorre um *hazard* quando existe a possibilidade de o circuito gerar um *glitch*.
- Ocorre um *hazard* **estático** quando existe a possibilidade de uma saída sofrer uma transição momentânea em condições em que se esperava que ela se mantivesse inalterada.
- Ocorre um *hazard* **dinâmico** quando for possível uma saída mudar mais do que uma vez, em condições em que se esperava que ela tivesse uma única transição (de 0 → 1 ou de 1→0).

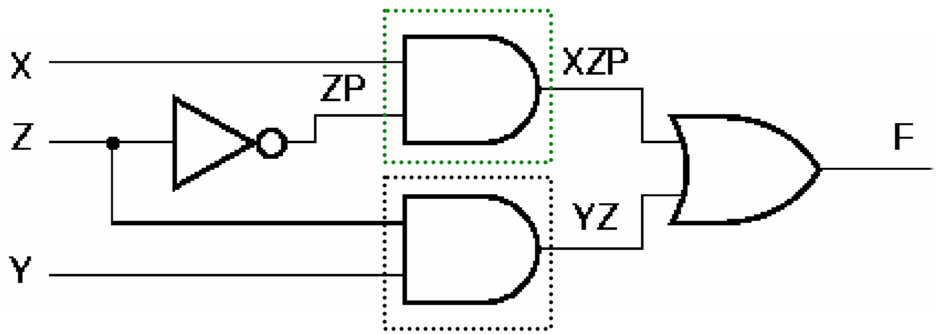


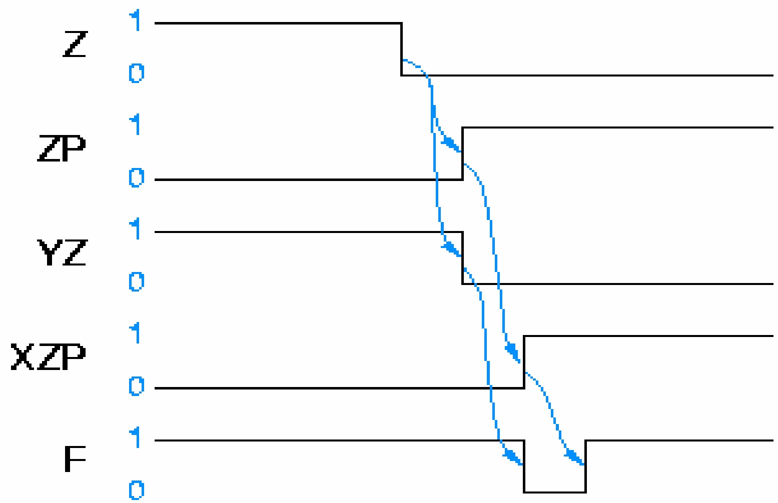
3. Conceitos sobre Sistemas Combinacionais

- Hazards (2) -

- Um *hazard* **num 1 (0) estático** é um par de combinações de entradas que diferem apenas numa variável de entrada e em que ambas produzem 1 (0) na saída, de tal modo que pode ocorrer um 0 (1) momentâneo na saída, durante uma transição na variável de entrada que distingue essas combinações.

- Exemplo: X=Y=1 e Z transita de 1 → 0



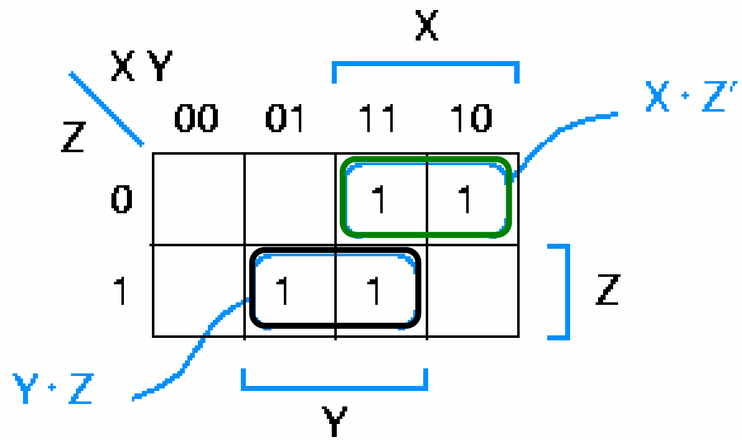


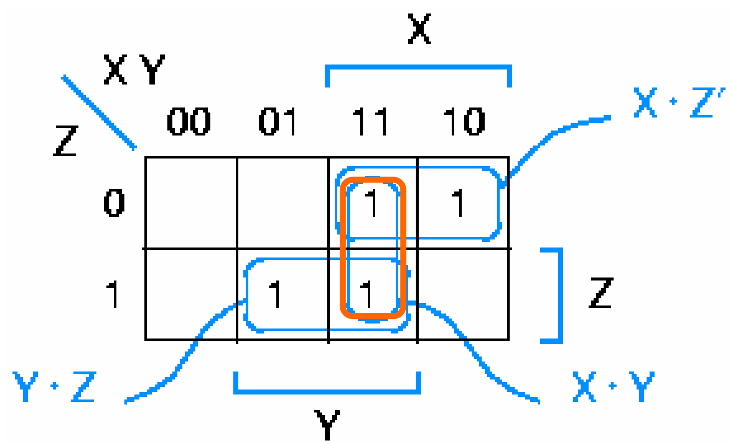
- Os métodos usados para eliminar *hazards* consideram que apenas uma entrada varia em cada instante. Este pressuposto equivale a efectuar um deslocamento através de células adjacentes num mapa de Karnaugh.

3. Conceitos sobre Sistemas Combinacionais

- Hazards (3) -

- Os mapas de Karnaugh podem ser usados para detectar *hazards* estáticos em circuitos com estrutura AND-OR ou OR-AND.
- Num circuito bem projectado e que implemente uma soma de produtos a 2-níveis só podem ocorrer *hazards* em 1's estáticos (não em 0's estáticos).
- Como não há um termo de produto que cobre ambas as combinações XYZ=111 e XYZ=110, é possível que se gere um breve *glitch* a 0 na saída (se o AND que muda para 0 o fizer antes do AND que muda para 1).
- Para eliminar o *hazard*, deve incluir-se no circuito uma porta AND extra.

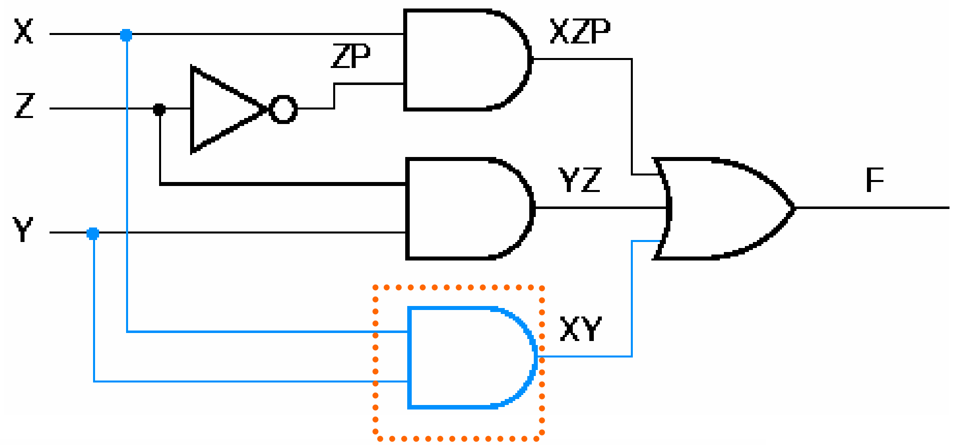


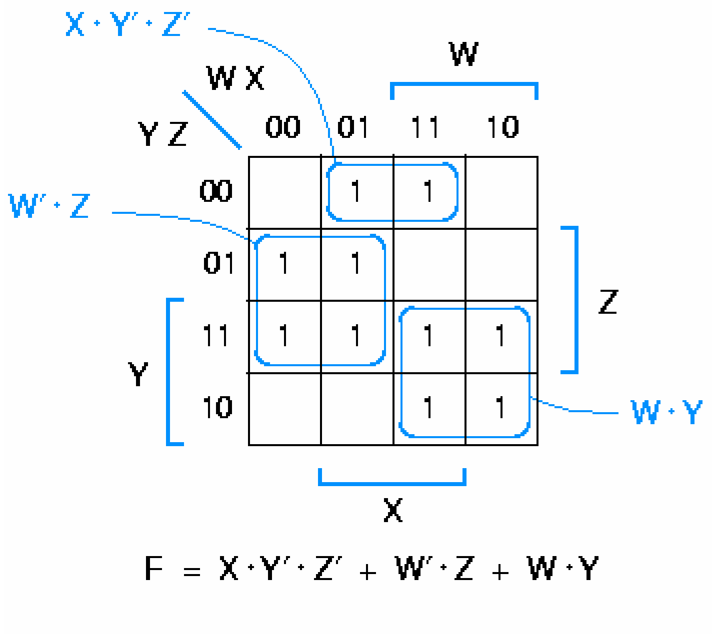


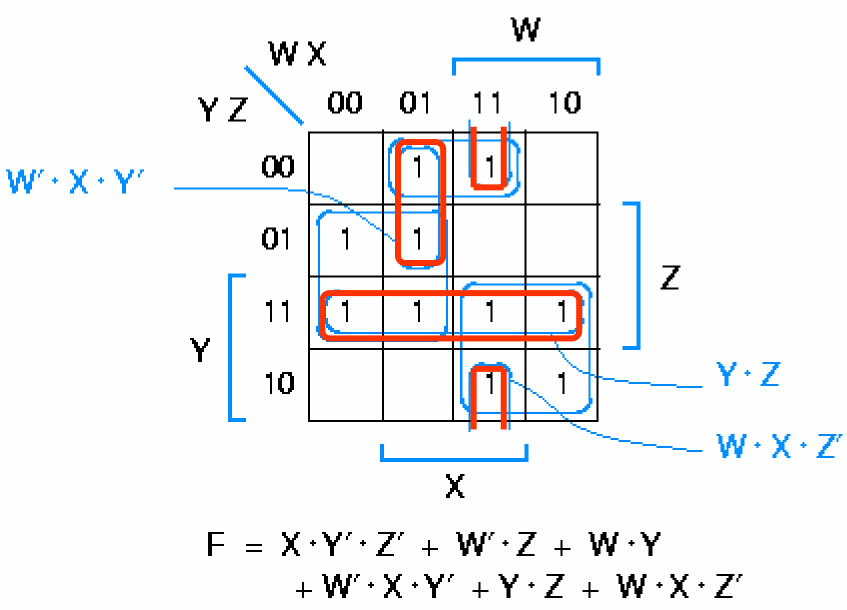
3. Conceitos sobre Sistemas Combinacionais

- *Hazards (4)* -

- Circuito resultante após eliminar o *hazard* com a porta AND extra
- Exemplo: 3 AND's extra para eliminar os *hazards estáticos*



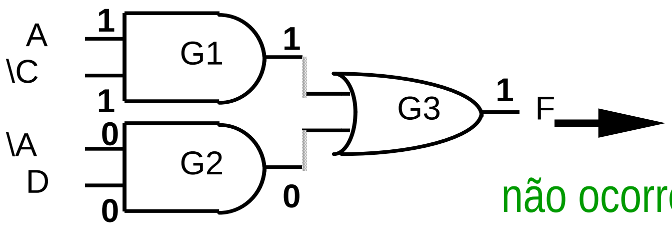




3. Conceitos sobre Sistemas Combinacionais

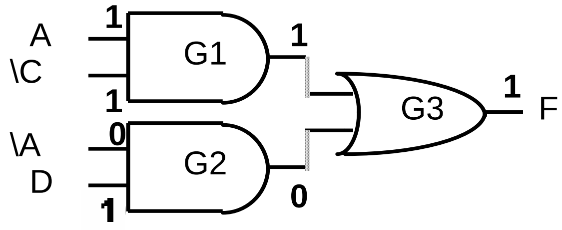
- Hazards (5) -

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

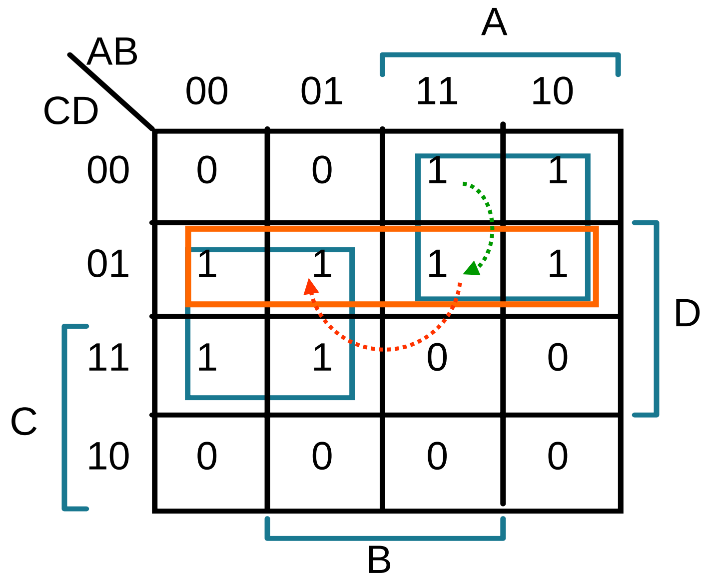


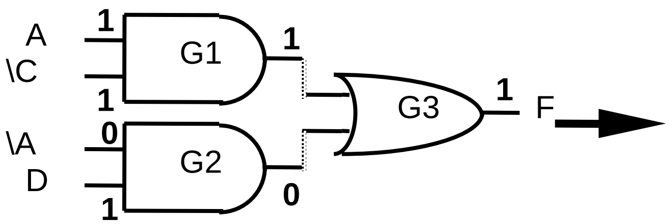
ABCD = 1100

não ocorre um glitch

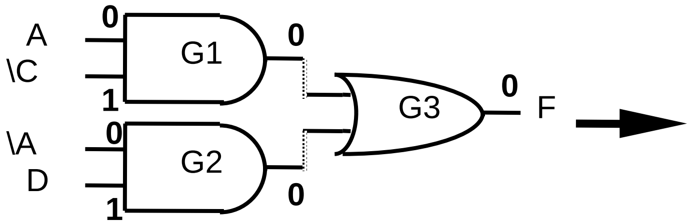


ABCD = 1101

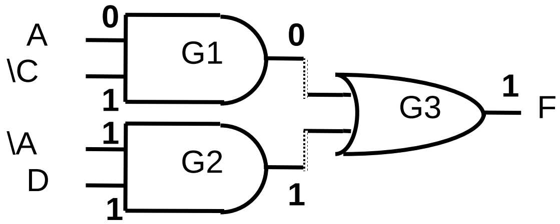




ABCD = 1101



ABCD = 0101 ($\overline{A}$ ainda é 0)



ABCD = 0101 ($\overline{A}$ já é 1)

ocorre um glitch → eliminado com o termo extra C'.D