ARCHITETTURA DEGLI ELABORATORI

A.A. 2020-2021

Università di Napoli Federico II

Corso di Laurea in Informatica

Docenti

Proff. Luigi Sauro gruppo 1 (A-G)

Silvia Rossi gruppo 2 (H-Z)

ALGEBRA DI BOOLE E RETI COMBINATORIE

Mappe di Karnaugh

- Le mappe di Karnaugh sono un metodo per semplificare espressioni booleane in forma SOP
- In realtà non introducono tecniche di semplificazione nuove, sono semplicemente un espediente grafico che consente di rilevare più facilmente implicati che possono essere semplificati
- Quindi alla base delle mappe di Karnaugh c'è il solito principio:

$$PA + P\overline{A} = P$$

Karnaugh Maps (K-Maps)

- Boolean expressions can be minimized by combining terms
- K-maps minimize equations graphically
- $PA + P\overline{A} = P$

A	B	C	Y
0	0	0	**1**
0	0	1	**1**
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Y: C \ AB	00	01	11	10
0	**1**	0	0	0
1	**1**	0	0	0

Y: C \ AB	00	01	11	10
0	$\mathbf{\bar{A}\bar{B}\bar{C}}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\mathbf{\bar{A}\bar{B}C}$	$\bar{A}BC$	ABC	$A\bar{B}C$

K-Map

- Circle 1's in adjacent squares
- In Boolean expression, include only literals whose true and complement form are ***not*** in the circle

A	B	C	Y
0	0	0	**1**
0	0	1	**1**
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Y

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

$$Y = \overline{A}\,\overline{B}$$

3-Input K-Map

Y (C \ AB)	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

Truth Table

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	**1**
0	1	1	**1**
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	**1**

K-Map

Y (C \ AB)	00	01	11	10
0				
1				

K-Map Definitions

- **Complement:** variable with a bar over it

  $\bar{A}, \bar{B}, \bar{C}$

- **Literal:** variable or its complement

  $\bar{A}, A, \bar{B}, B, C, \bar{C}$

- **Implicant:** product of literals

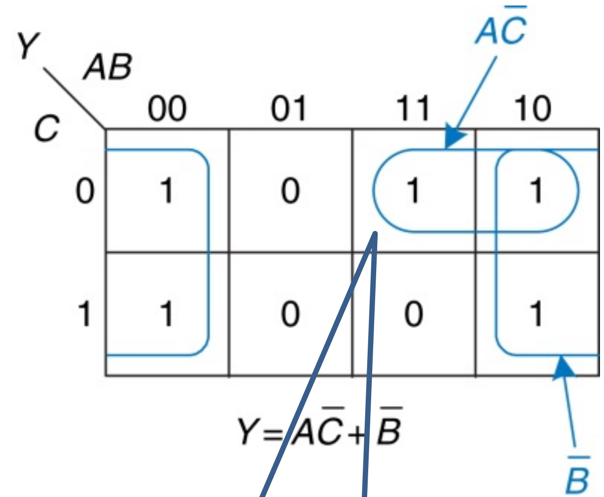
  $A\bar{B}C, \bar{A}C, BC$

- **Prime implicant:** implicant corresponding to the largest circle in a K-map

K-Map Rules

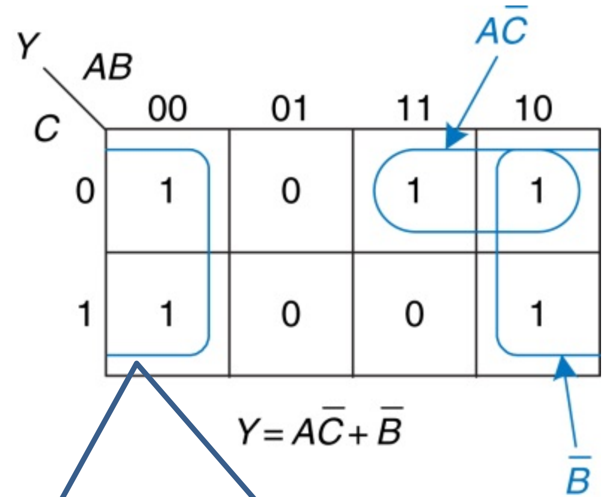
- Every 1 must be circled at least once
- Each circle must span a power of 2 (i.e. 1, 2, 4) squares in each direction
- Each circle must be as large as possible
- A circle may wrap around the edges
- A “don't care” (X) is circled only if it helps minimize the equation

Y (C \ AB)	00	01	11	10
0	1	0	1	1
1	1	0	0	1



$AB\bar{C} + A\bar{B}\bar{C} = A\bar{C}(B + \bar{B}) = A\bar{C}$

Y: C \ AB	00	01	11	10
0	1	0	1	1
1	1	0	0	1



$$\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C = \bar{A}\bar{B}(\bar{C} + C) + A\bar{B}\,(\bar{C} + C) = \bar{A}\bar{B} + A\bar{B} = (\bar{A} + A)\,\bar{B} = \bar{B}$$

4-Input K-Map

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y

CD \ AB	00	01	11	10
00				
01				
11				
10				

4-Input K-Map

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y

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

4-Input K-Map

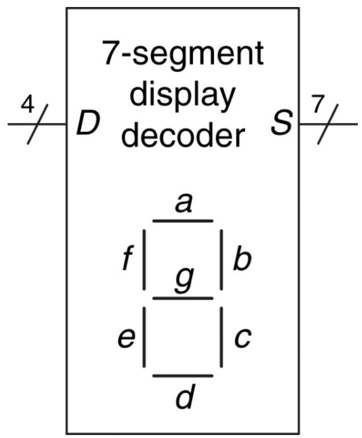
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y

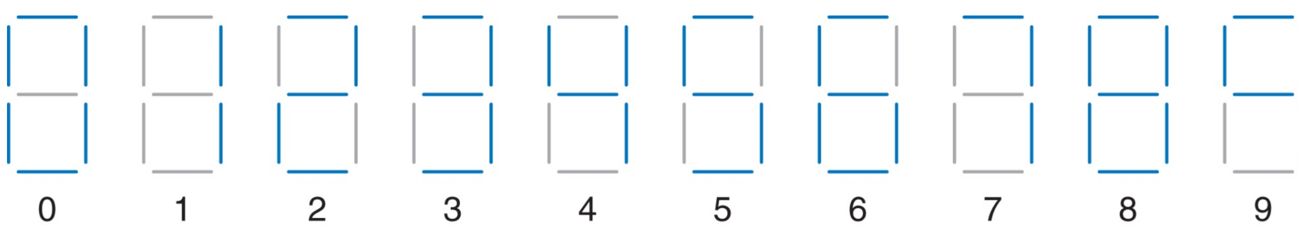
CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

$$Y = \underbrace{\overline{A}C}_{3} + \underbrace{\overline{A}BD}_{1} + \underbrace{A\overline{B}\,\overline{C}}_{2} + \underbrace{\overline{B}\,\overline{D}}_{4}$$

Display a 7 segmenti



$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0



Display a 7 segmenti

$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0

Display a 7 segmenti

S_a

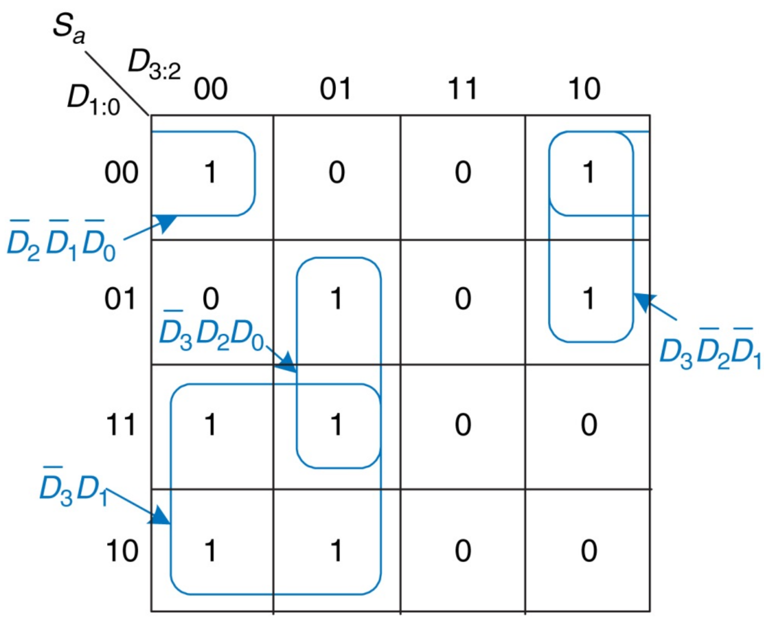
$D_{1:0}$ \ $D_{3:2}$	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	0

S_b

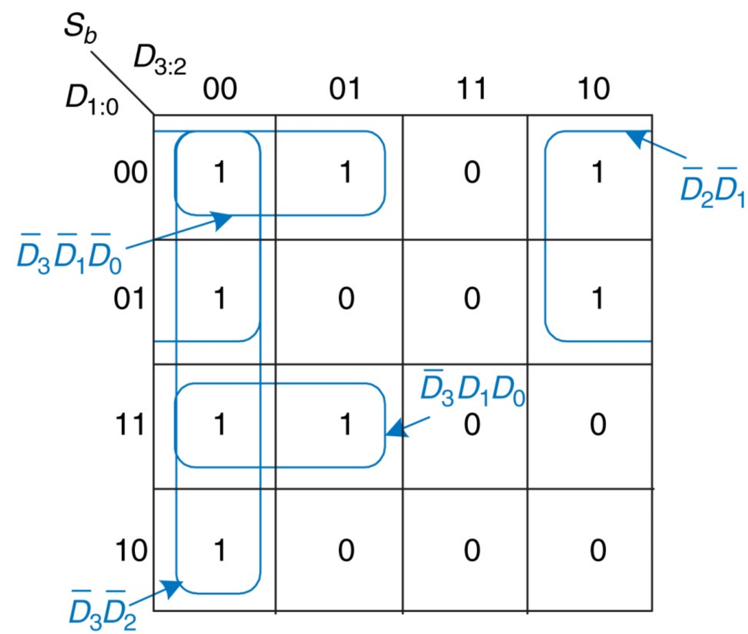
$D_{1:0}$ \ $D_{3:2}$	00	01	11	10
00	1	1	0	1
01	1	0	0	1
11	1	1	0	0
10	1	0	0	0

$\overline{D}_2\overline{D}_1\overline{D}_0 + D_3\overline{D}_2\overline{D}_1 + \overline{D}_3 D_2 D_0 + \overline{D}_3 D_1$

Display a 7 segmenti



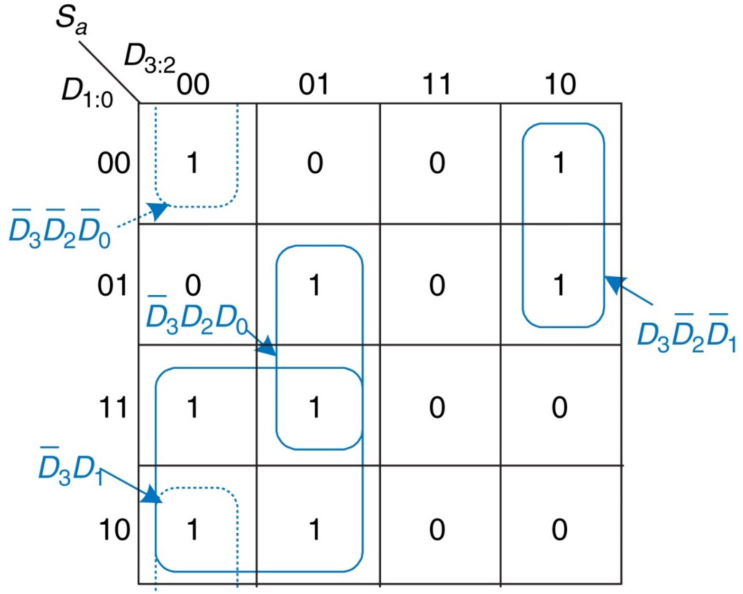
$S_a = \overline{D}_3D_1 + \overline{D}_3D_2D_0 + D_3\overline{D}_2\overline{D}_1 + \overline{D}_2\overline{D}_1\overline{D}_0$



$S_b = \overline{D}_3\overline{D}_2 + \overline{D}_2\overline{D}_1 + \overline{D}_3D_1D_0 + \overline{D}_3\overline{D}_1\overline{D}_0$

Display a 7 segmenti

In generale, vi possono essere diverse forme minime:



$$S_a = \overline{D}_3D_1 + \overline{D}_3D_2D_0 + D_3\overline{D}_2\overline{D}_1 + \overline{D}_3\overline{D}_2\overline{D}_0$$

Don’t care in output

- Abbiamo visto nei circuiti a priorità come si possano utilizzare degli input «don’t care» per avere una rappresentazione della tabella di verità più succinta
- Valori «don’t care» (X) si possono avere anche in output allorché per una data configurazione degli input il valore dell’output è irrilevante (non ci interessa se sia 0 o 1)
- Ad esempio nel display a sette segmenti per gli input «illegali» 10-15 l’output può essere di tipo X
- Poiché non ci interessa se un valore X corrisponde in realtà ad uno 0 o un 1 allora nella minimizzazione di una espressione possiamo scegliere che valore dargli come più opportuno.
- In particolare in una K-map sostituiremo i valori X con degli 1 se questo consente di avere un numero inferiore di cerchi o cerchi più larghi

K-Maps with Don't Cares

A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y

CD \ *AB*	00	01	11	10
00				
01				
11				
10				

K-Maps with Don't Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y

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

K-Maps with Don’t Cares

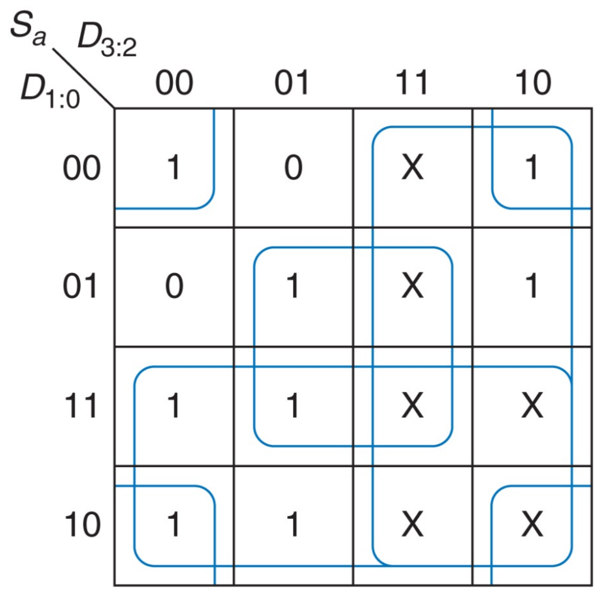
A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y

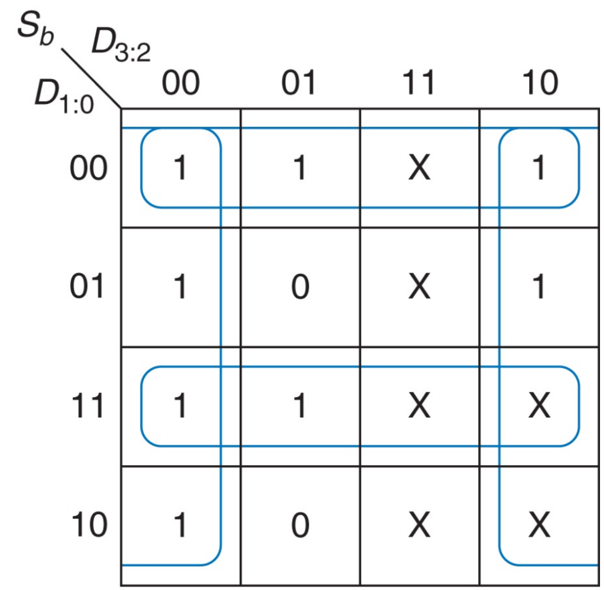
CD \ *AB*	00	01	11	10
00	1	0	X	1
01	0	X	X	1
11	1	1	X	X
10	1	1	X	X

$$Y = A + \overline{B}\,\overline{D} + C$$

K-map con valori «don't care»



$S_a = D_3 + D_2 D_0 + \overline{D}_2 \overline{D}_0 + D_1$



$S_b = \overline{D}_2 + D_1 D_0 + \overline{D}_1 \overline{D}_0$

Esercizi su K-maps

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

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

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

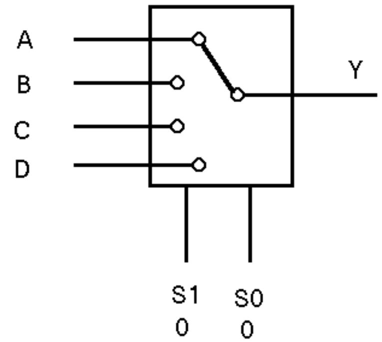
AB CD	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

$1 = A\bar{B} + BD + \bar{B}\bar{D}$

$2 = CD + \bar{A}D + \bar{B}$

Multiplexer

- Un multiplexer è sostanzialmente un selettore di linea



- In generale è costituito da N ingressi (dove N è una potenza di 2), 1 uscita, e $\log_2 N$ linee di selezione che indicano a quale ingresso deve corrispondere l'output

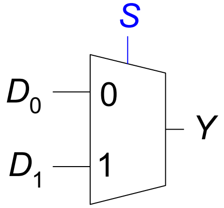
Combinational Building Blocks

- Multiplexers
- Decoders

Multiplexer (Mux)

- Selects between one of *N* inputs to connect to output
- $\log_2 N$-bit select input – control input
- **Example:**

2:1 Mux

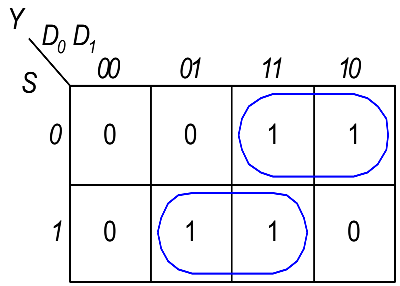


S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

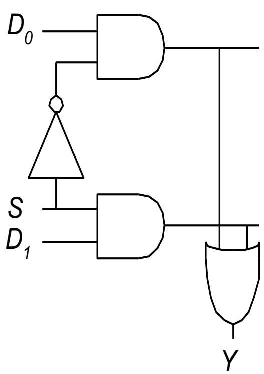
S	Y
0	D_0
1	D_1

Multiplexer Implementations

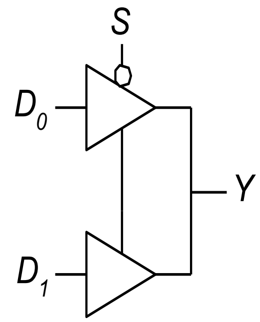
- **Logic gates**
  - Sum-of-products form



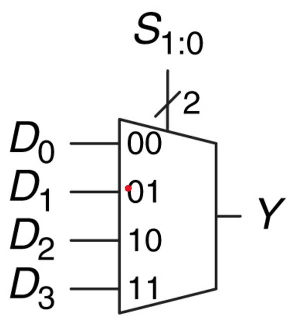
$Y = D_0\overline{S} + D_1 S$

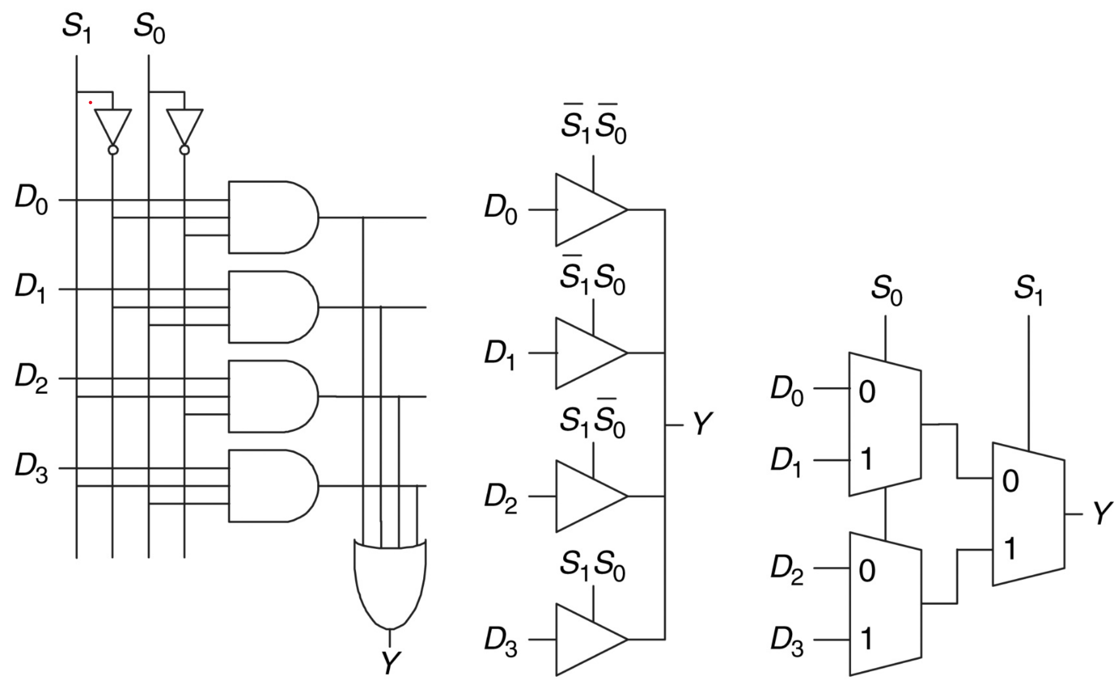


- **Tristates**
  - For an N-input mux, use N tristates
  - Turn on exactly one to select the appropriate input



Multiplexer 4:1





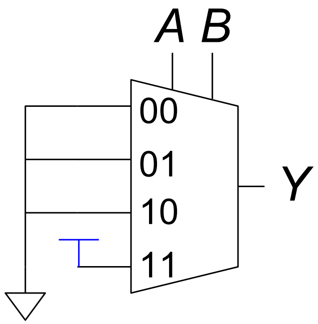
(a) **(b)** **(c)**

Logic using Multiplexers

Using mux as a lookup table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$$Y = AB$$



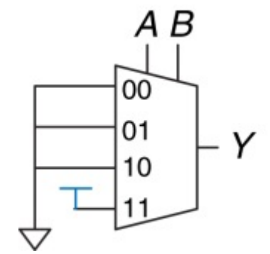
Sintetizzare funzioni booleane con mux

- I multiplexer possono essere usati anche per sintetizzare delle funzioni booleane
- Sintetizzare una funzione di m variabili con un mux a 2^m linee è molto semplice: le variabili saranno linee di selezione. Data una certa configurazione delle variabili, la linea di ingresso corrispondente sarà posta al valore della funzione in quella configurazione
- Di fatto le linee di ingresso riproducono la tabella di verità

  della funzione

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$

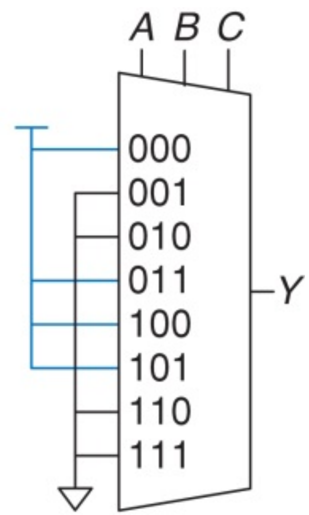


Sintetizzare funzioni booleane con mux

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$Y = A\overline{B} + \overline{B}\,\overline{C} + \overline{A}BC$

(a)



(b)

$Y = \Sigma(0,3,4,5)$

Sintetizzare funzioni booleane con mux

- E’ possibile utilizzare un mux con 2^{m-1} ingressi per sintetizzare un funzione ad m variabili
- Le prime m-1 variabili saranno linee di selezione, mentre le linee di ingresso possono essere poste a 0,1, oppure all’ultima variabile (positiva o negata)

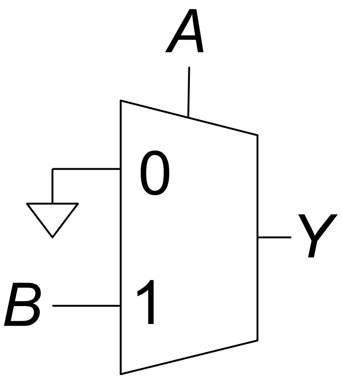
Logic using Multiplexers

Reducing the size of the mux

$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	Y
0	0
1	B



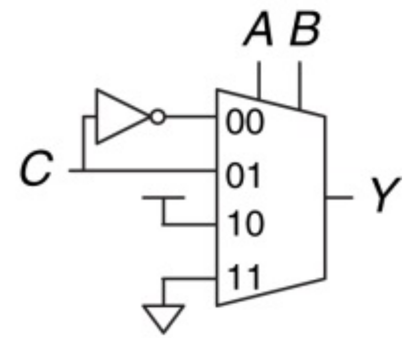
Sintetizzare funzioni booleane con mux

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(a)

A	B	Y
0	0	$\overline{C}$
0	1	C
1	0	1
1	1	0

(b)



(c)