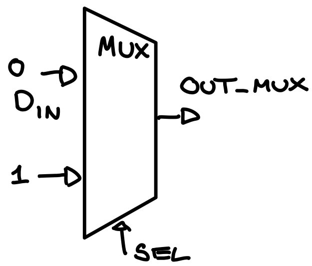
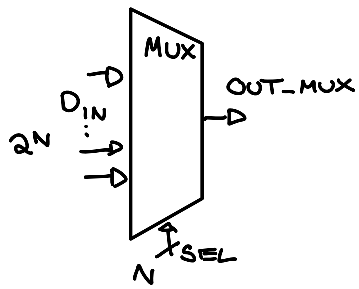
→ Multiplexer Symbol



2-INPUT MUX



General MUX

Signal	Direction	Resolution
D_{IN}	IN	N_D
SEL	IN	$\log_2(N_D)$
OUT_MUX	OUT	1

"ho quindi una word di n ingressi e scelgo quale bit leggere."

"I Multiplexers selezionano uno specifico segnale di bit/bus di ingresso e forniscono tale segnale in uscita."

PSEUDO-CODICE

```
if SEL=0
 then
 OUT_MUX= DIN(0)
 ELSE   OUT_MUX = DIN(1)
end IF;
```

⚠ sel è il numero dell'ingresso da leggere
↓
può essere formato da più ingressi
↓
bit per il numero

Tabella di verità

DIN(1)	DIN(0)	SEL	OUT MAX
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Kmap

OUT_MUX		DIN(1) DIN(0) 00	01	11	10
SEL	0	0	1	1	0
	1	0	0	1	1

OUT_MUX =

$\overline{SEL} \cdot DIN(0)$

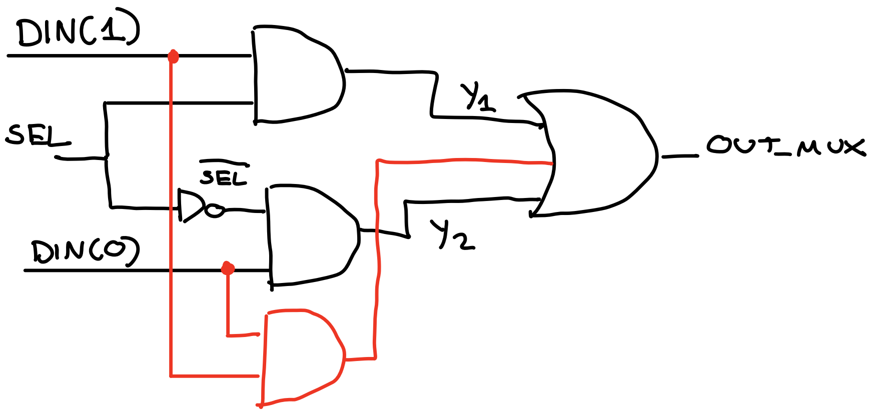
$+$

$SEL \cdot DIN(1)$

$+$

qualsiasi sia SEL esce 1 → $DIN(0) \cdot DIN(1)$

Schema



MUX - 1x2

$$OUT\_MUX = SEL \cdot D_1 + \overline{SEL}\, D_0$$

MUX - 2x4

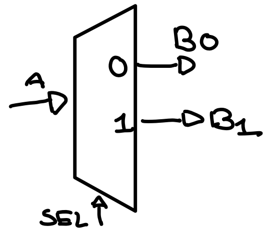
$$OUT\_MUX = SEL(1) \cdot SEL(0) \cdot D_3 + SEL(1) \cdot \overline{SEL}(0) \cdot D_2 + \overline{SEL}(1) \cdot SEL(0) \cdot D_1 + \overline{SEL}(1) \cdot \overline{SEL}(0)\, D_0$$

→ posso ricostruirlo utilizzando 3 MUX 1x2 !

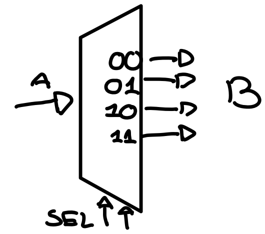
→ $N \times 2^N$ MUX può essere costruito utilizzando $2^N - 1$ 1x2 MUX

→ **Demultiplexer Symbol**

DEMUX 2x1



DEMUX 4x2

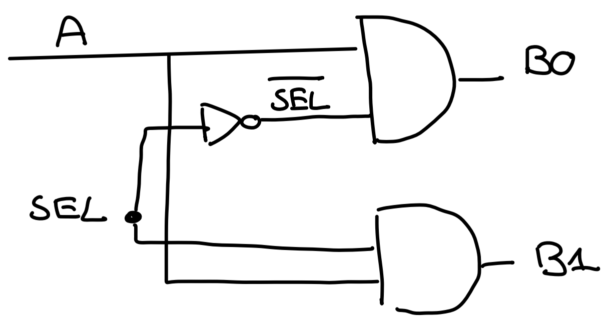


Pseudo-codice

```
If SEL=0
then
 B0=A   else  B1=A
 end if;
```

INPUT		OUTPUT	
A	SEL	B0	B1
0	0	0	x
0	1	x	0
1	0	1	x
1	1	x	1

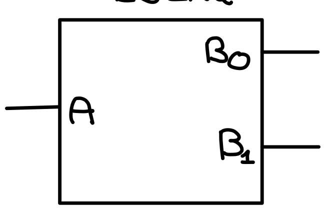
non definito



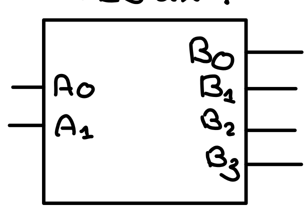
Azione contraria del multiplexer.

→ Decoder

DEC 1x2



DEC 2x4



Signal	Direction	Resolution
A	In	$\log_2(Nb)$
B	Out	Nb

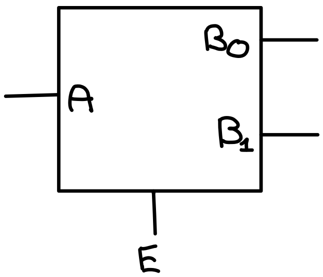
Codice per DEC 1x2

```
if A=0 then
    B0 = 1
    B1 = 0
 else
    B0 = 0
    B1 = 1
 endif;
```

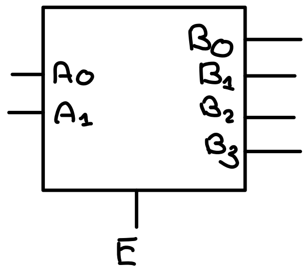
INPUT		OUTPUT			
A1	A0	B3	B2	B1	B0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

–▷ Decoder con Enable

DEC 1x2



DEC 2x4



Signal	Direction	Resolution
A	In	$\log_2(N_b)$
B	Out	Nb
E	In	1

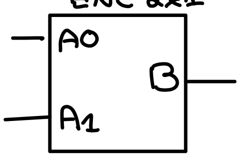
Pseudo-code

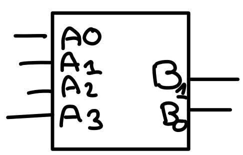
```
if  enable = 1  then
        = al precede
else   B0=0 ; B1=0;
end if;
```

INPUT			OUTPUT			
ENABLE	A1	A0	B3	B2	B1	B0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

–▷ Encoder

ENC 2x1





Signal	Direction	Resolution
A	In	Nb
B	Out	$\log_2(N_b)$

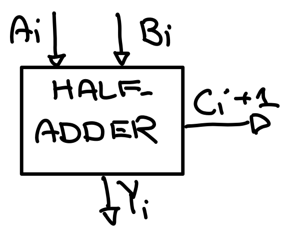
Pseudo-code ENC 4x2

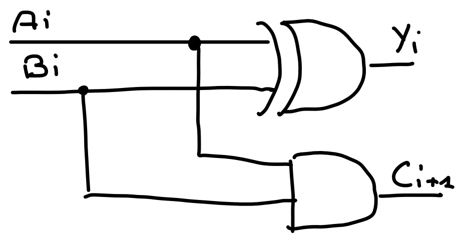
```
Switch A
    case '00' : B=X
    case '01' : B=0
    case '10' : B=1
    case '11' : B=x
end
```

INPUT				OUTPUT	
A3	A2	A1	A0	B1	B0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

⚠ Contrario del Multiplexer!

–▷ Half-Adder





A_i	B_i	C_{i+1}	Y_i
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Esercizio 1

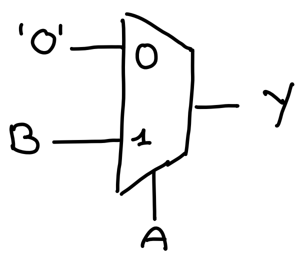
"Sintetizzare logicamente AND, OR, NAND, NOR e XOR usando 1x2-MUX e INVERTER."

- LOGIC AND

$A = SEL$

$$Y = A \cdot B = \begin{cases} 0 & \text{if } SEL = '0' \\ B & \text{if } SEL = '1' \end{cases}$$

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

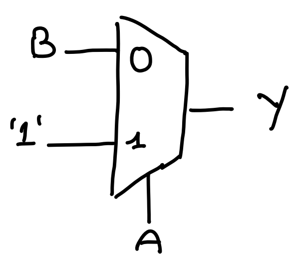


- LOGIC OR

$A = SEL$

$$Y = A + B = \begin{cases} B & \text{if } SEL = '0' \\ 1 & \text{if } SEL = '1' \end{cases}$$

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

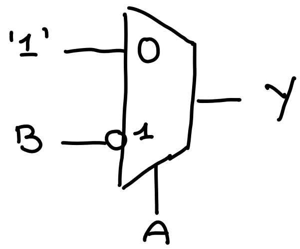


- LOGIC NAND

$A = SEL$

$$Y = \overline{A \cdot B} = \begin{cases} 1 & \text{if } SEL = '0' \\ \overline{B} & \text{if } SEL = '1' \end{cases}$$

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

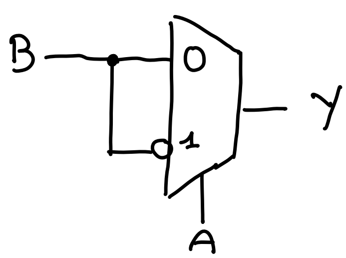


- LOGIC XOR

$A = SEL$

$$Y = A \oplus B = \begin{cases} B & \text{if } SEL = '0' \\ \overline{B} & \text{if } SEL = '1' \end{cases}$$

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

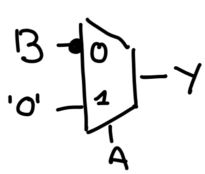


- LOGIC NOR

$A = SEL$

$$Y = \overline{A + B} = \begin{cases} \overline{B} & \text{se } A = 0 \\ 0 & \text{se } A = 1 \end{cases}$$

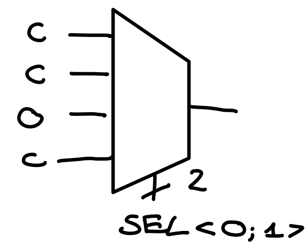
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



Esercizio 2

Sintetizzare con 2x4 MUX e Inverter la seguente funzione logica:

$$W_1 = A \cdot B \cdot C + \bar{A}\bar{B}C + \bar{A} \cdot B \cdot \bar{C}$$



Generic MUX - 2x4

Fix SEL<1> = A   SEL<0> = B

W1 può essere scritta come

$$W_1 = \underbrace{D_0}_{C} \cdot \bar{A} \cdot \bar{B} + \underbrace{D_1}_{\bar{C}} \cdot \bar{A} \cdot B + \underbrace{D_2}_{0} \cdot A \cdot \bar{B} + \underbrace{D_3}_{C} \cdot A \cdot B$$

Sel = BA → 00, 01, 10, 11

W1

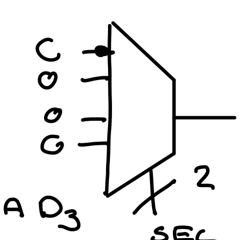
$$\underbrace{\overset{\bar{A}\bar{B}}{00}}_{\text{guardo } W_1} \underbrace{D_0}_{C} + \overset{B\bar{A}}{10}\, \underbrace{D_1}_{\bar{C}} + \overset{\bar{B}A}{01}\, \underbrace{D_2}_{0} + \overset{AB}{11} \cdot \underbrace{D_3}_{C}$$

non c'è!

Esercizio 3

Sintetizzare con 2x4 MUX e Inverter la seguente funzione logica:

$$W_1 = A \cdot B \cdot C + \bar{A}\bar{B}\bar{C}$$



SEL<0> = A

$$\overline{BA}\, \underbrace{D_0}_{\bar{C}} + B\bar{A}\, \underbrace{D_1}_{0} + \bar{B}A\, \underbrace{D_2}_{0} + BA\, \underbrace{D_3}_{C}$$

Esercizio 4

Design un prime Number Detector (PND) usando un 3x8 MUX e INVERTERs

① Facciamo la tabella di verità!

Int	A_3	A_2	A_1	A_0	Y
0	0	0	0	0	0
1	0	0	0	1	0
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
⋮	⋮	⋮	⋮	⋮	⋮
11	1	0	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮
13	1	1	0	1	1
⋮	⋮	⋮	⋮	⋮	

② Scrivo la funzione Y Non minimizzata

$$Y = \underbrace{\overline{A3}\cdot\overline{A2}\cdot A1\cdot\overline{A0}}^{(2)} + \overbrace{\overline{A3}\cdot\overline{A2}\cdot A1\cdot A0}^{(3)} +$$

$$+ \overbrace{\overline{A3}\cdot A2\cdot\overline{A1}\cdot A0}^{(5)} + \overbrace{\overline{A3}\cdot A2\cdot A1\cdot A0}^{(7)} +$$

$$+ \overbrace{A3\cdot\overline{A2}\cdot A1\cdot A_0}^{(11)} + \overbrace{A3\, A2\, \overline{A1}\, A0}^{(13)}$$

③ Minimizziamdo

$$Y = \overline{A3}\cdot\overline{A2}\cdot A1 + \overline{A3}\cdot A2\cdot A0 + A3\cdot\overline{A2}\cdot A1\cdot A0 +$$

$$+ A3\, A2\, \overline{A1}\, A0$$

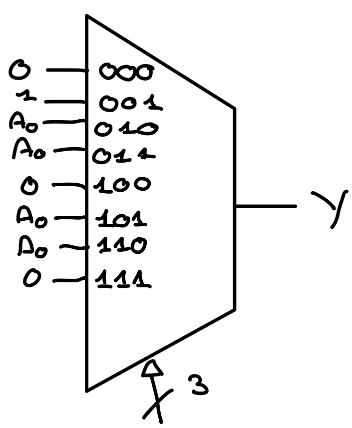
④ Tabella

Int	Selection Word Code	Di	Value
0 1	$\overline{A_3}\cdot\overline{A_2}\cdot\overline{A_1}$	D_{000}	0
2 3	$\overline{A_3}\;\overline{A_2}\cdot A_1$	D_{001}	1
4 5	$\overline{A_3}\cdot A_2\;\overline{A_1}$	D_{010}	A_0
⋮	⋮	⋮	⋮

MUX → dove c'é 1 o A_0

$$
\begin{aligned}
Y = {} & \overline{A_3}\cdot\overline{A_2}\cdot A_1\cdot D_{001} + \overline{A_3}\cdot A_2\cdot\overline{A_1}\cdot D_{010} + \\
& + \overline{A_3}\cdot A_2\cdot A_1\cdot D_{011} + A_3\cdot\overline{A_2}\cdot A_1\cdot D_{101} + \\
& + A_3\cdot A_2\cdot\overline{A_1}\cdot D_{110}\,.
\end{aligned}
$$

⑤



SEL<0:2> = <$A_3\,A_2\,A_1$>

RICORDO

Nella mappa di K-Map il passaggio in diagonale è inutile