Le basi matematiche dell'Informatica

Alfredo Marzocchi

Università Cattolica del Sacro Cuore
Dipartimento di Matematica e Fisica "Niccolò Tartaglia"
Via dei Musei, 41 – 25121 Brescia (Italy)

La Matematica sta alla base dell'Informatica.

La Matematica sta alla base dell'Informatica.

Ma non nel senso che per usare Internet, Facebook o, più in generale, serva la Matematica...

La Matematica sta alla base dell'Informatica.

Ma non nel senso che per usare Internet, Facebook o, più in generale, serva la Matematica...

In realtà molta Matematica sta proprio alla *radice* dell'Informatica, ma così in profondità che nemmeno ce ne accorgiamo.

L'informazione

Siamo costantemente immersi in un flusso di *informazioni*, specie a scuola.

L'informazione

Siamo costantemente immersi in un flusso di *informazioni*, specie a scuola.

Per esempio, da quando avete cominciato a leggere queste slides avete letto

L'informazione

Siamo costantemente immersi in un flusso di *informazioni*, specie a scuola.

Per esempio, da quando avete cominciato a leggere queste slides avete letto

- 5 frasi (titoli esclusi);

L'informazione

Siamo costantemente immersi in un flusso di *informazioni*, specie a scuola.

Per esempio, da quando avete cominciato a leggere queste slides avete letto

- 5 frasi (titoli esclusi);
- 68 parole (titoli esclusi).

L'informazione

Siamo costantemente immersi in un flusso di *informazioni*, specie a scuola.

Per esempio, da quando avete cominciato a leggere queste slides avete letto

- 5 frasi (titoli esclusi);
- 68 parole (titoli esclusi).

Una "parola" è un *elemento di informazione*. Essa racchiude un significato, anche se spesso una parola da sola non significa nulla.

L'informazione

Siamo costantemente immersi in un flusso di *informazioni*, specie a scuola.

Per esempio, da quando avete cominciato a leggere queste slides avete letto

- 5 frasi (titoli esclusi);
- 68 parole (titoli esclusi).

Una "parola" è un *elemento di informazione*. Essa racchiude un significato, anche se spesso una parola da sola non significa nulla. Allo stesso modo, i singoli *caratteri* che compongono la parola (lettere, segni di punteggiatura, spazi) non hanno significato da soli ma lo acquistano quando diventano parole.

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

...

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

...

Volume

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

...
Volume
Capitolo

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

...
Volume
Capitolo
Paragrafo

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

...
Volume
Capitolo
Paragrafo
Capoverso

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)
Parola

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)
Parola
Lettera

All'estremo opposto, tante frasi formano un capoverso, tanti capoversi un paragrafo, tanti paragrafi un capitolo, tanti capitoli un volume, ecc.
Abbiamo allora una "gerarchia di complessità"

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)
Parola
Lettera
... ?

È chiaro che verso l'alto possiamo aggiungere strutture sempre più complesse, ma verso il basso cosa possiamo trovare?

È chiaro che verso l'alto possiamo aggiungere strutture sempre più complesse, ma verso il basso cosa possiamo trovare? Esiste qualcosa di più semplice di una lettera?

È chiaro che verso l'alto possiamo aggiungere strutture sempre più complesse, ma verso il basso cosa possiamo trovare? Esiste qualcosa di più semplice di una lettera?
Consideriamo questa frase (mettiamo tutto minuscolo per semplicità):

È chiaro che verso l'alto possiamo aggiungere strutture sempre più complesse, ma verso il basso cosa possiamo trovare? Esiste qualcosa di più semplice di una lettera?
Consideriamo questa frase (mettiamo tutto minuscolo per semplicità):

roma è la capitale d'italia.

È chiaro che verso l'alto possiamo aggiungere strutture sempre più complesse, ma verso il basso cosa possiamo trovare? Esiste qualcosa di più semplice di una lettera?
Consideriamo questa frase (mettiamo tutto minuscolo per semplicità):

roma è la capitale d'italia.

Numeriamo ora le lettere dell'alfabeto italiano e i segni di punteggiatura in qualche modo (tanto è solo un esempio):

È chiaro che verso l'alto possiamo aggiungere strutture sempre più complesse, ma verso il basso cosa possiamo trovare? Esiste qualcosa di più semplice di una lettera?
Consideriamo questa frase (mettiamo tutto minuscolo per semplicità):

roma è la capitale d'italia.

Numeriamo ora le lettere dell'alfabeto italiano e i segni di punteggiatura in qualche modo (tanto è solo un esempio):

a	b	c	d	e	f	g	h	i	l
01	02	03	04	05	06	07	08	09	10
m	n	o	p	q	r	s	t	u	v
11	12	13	14	15	16	17	18	19	20
z	è	,	.	;	:	?	!	'	(spazio)
21	22	23	24	25	26	27	28	29	30

È chiaro che verso l'alto possiamo aggiungere strutture sempre più complesse, ma verso il basso cosa possiamo trovare? Esiste qualcosa di più semplice di una lettera?
Consideriamo questa frase (mettiamo tutto minuscolo per semplicità):

roma è la capitale d'italia.

Numeriamo ora le lettere dell'alfabeto italiano e i segni di punteggiatura in qualche modo (tanto è solo un esempio):

a	b	c	d	e	f	g	h	i	l
01	02	03	04	05	06	07	08	09	10
m	n	o	p	q	r	s	t	u	v
11	12	13	14	15	16	17	18	19	20
z	è	,	.	;	:	?	!	'	(spazio)
21	22	23	24	25	26	27	28	29	30

(abbiamo dovuto aggiungere qualcosa—la 'è'— altrimenti non sapevamo come fare)

Questo è un esempio (molto rozzo) di *codifica*. Proviamo ora a tradurre la nostra frase facendo uso della tabella:

Questo è un esempio (molto rozzo) di *codifica*. Proviamo ora a tradurre la nostra frase facendo uso della tabella:

roma è la capitale d'italia.

Questo è un esempio (molto rozzo) di *codifica*. Proviamo ora a tradurre la nostra frase facendo uso della tabella:

roma è la capitale d'italia.

a	b	c	d	e	f	g	h	i	l
01	02	03	04	05	06	07	08	09	10
m	n	o	p	q	r	s	t	u	v
11	12	13	14	15	16	17	18	19	20
z	è	,	.	;	:	?	!	'	(spazio)
21	22	23	24	25	26	27	28	29	30

Questo è un esempio (molto rozzo) di *codifica*. Proviamo ora a tradurre la nostra frase facendo uso della tabella:

roma è la capitale d'italia.

a	b	c	d	e	f	g	h	i	l
01	02	03	04	05	06	07	08	09	10
m	n	o	p	q	r	s	t	u	v
11	12	13	14	15	16	17	18	19	20
z	è	,	.	;	:	?	!	'	(spazio)
21	22	23	24	25	26	27	28	29	30

r	o	m	a	(spazio)	è	(spazio)	l	a	(spazio)	c...
16	13	11	01	30	22	30	10	01	30	03...

Il risultato finale è

Il risultato finale è

$$1613110130223010013003011409180110053004290918011024.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.

Il risultato finale è

$$1613110130223010013003011409180110053004290918011024.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

Il risultato finale è

$$1613110130223010013003011409180110053004290918011024.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

...

Il risultato finale è

$$1613110130223010013003011409180110053004290918011024.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

...

Volume

Il risultato finale è

$$1613110130223010013003011409180110053004290918011024.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

...
Volume
Capitolo

Il risultato finale è

$$1613110130223010013003011409180110053004290918011024.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

...
Volume
Capitolo
Paragrafo

Il risultato finale è

$$161311013022301001300301140918011005300429091801 1024.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

...
Volume
Capitolo
Paragrafo
Capoverso

Il risultato finale è

$$161311013022301001300301140918011005300429091801102 4.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)

Il risultato finale è

$$161311013022301001300301140918011005300429091801102 4.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)
Parola

Il risultato finale è

$$1613110130223010013003011409180110053004290918011024.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)
Parola
Lettera

Il risultato finale è

$$1613110130223010013003011409180110053004290918011024.$$

Osserviamo che non abbiamo perso informazioni: se possediamo la tabella, possiamo ricostruire la frase in maniera 'leggibile'.
A questo punto sembra che vi sia qualcosa di ancora più elementare della 'lettera': la *cifra* da 0 a 9:

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)
Parola
Lettera
Cifra (0-9)
... ?

Possiamo andare ancora più oltre?

Possiamo andare ancora più oltre? Sì, usando la *numerazione binaria* dei numeri da 0 a 9, che riportiamo:

Possiamo andare ancora più oltre? Sì, usando la *numerazione binaria* dei numeri da 0 a 9, che riportiamo:

0	1	2	3	4	5	6	7	8	9
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Possiamo andare ancora più oltre? Sì, usando la *numerazione binaria* dei numeri da 0 a 9, che riportiamo:

0	1	2	3	4	5	6	7	8	9
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Guardate che fine fa allora la nostra frase:

Possiamo andare ancora più oltre? Sì, usando la *numerazione binaria* dei numeri da 0 a 9, che riportiamo:

0	1	2	3	4	5	6	7	8	9
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Guardate che fine fa allora la nostra frase:

r		o		m		a...	
1	6	1	3	1	1	0	1...
0001	0110	0001	0010	0001	0001	0000	0001...

Possiamo andare ancora più oltre? Sì, usando la *numerazione binaria* dei numeri da 0 a 9, che riportiamo:

0	1	2	3	4	5	6	7	8	9
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Guardate che fine fa allora la nostra frase:

r		o		m		a...	
1	6	1	3	1	1	0	1...
0001	0110	0001	0010	0001	0001	0000	0001...

È diventata lunghissima:

Possiamo andare ancora più oltre? Sì, usando la *numerazione binaria* dei numeri da 0 a 9, che riportiamo:

0	1	2	3	4	5	6	7	8	9
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Guardate che fine fa allora la nostra frase:

r		o		m		a...	
1	6	1	3	1	1	0	1...
0001	0110	0001	0010	0001	0001	0000	0001...

È diventata lunghissima:

16131101302230100130030114091801100530042909180110 24

Possiamo andare ancora più oltre? Sì, usando la *numerazione binaria* dei numeri da 0 a 9, che riportiamo:

0	1	2	3	4	5	6	7	8	9
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Guardate che fine fa allora la nostra frase:

r		o		m		a...	
1	6	1	3	1	1	0	1...
0001	0110	0001	0010	0001	0001	0000	0001...

È diventata lunghissima:

16131101302230100130030114091801100530042909180110 24

diventa

Possiamo andare ancora più oltre? Sì, usando la *numerazione binaria* dei numeri da 0 a 9, che riportiamo:

0	1	2	3	4	5	6	7	8	9
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Guardate che fine fa allora la nostra frase:

r		o		m		a...	
1	6	1	3	1	1	0	1...
0001	0110	0001	0010	0001	0001	0000	0001...

È diventata lunghissima:

16131101302230100130030114091801100530042909180110 24

diventa

000101100001001000010001000000010010000000100010001000 00...

(non ci sta nemmeno tutta...)

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

...

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

...
Volume

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

...
Volume
Capitolo

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

...
Volume
Capitolo
Paragrafo

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

...
Volume
Capitolo
Paragrafo
Capoverso

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)
Parola

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)
Parola
Lettera

Però, sapendo che le cifre vanno prese *a quattro a quattro*, possiamo ricostruire il messaggio quando vogliamo.
Quindi, con soli *due simboli* (0 e 1) siamo riusciti a trasmettere dell'informazione:

...
Volume
Capitolo
Paragrafo
Capoverso
Frase (periodo)
Parola
Lettera
Cifra (0-9)
Simbolo 0-1

Vien naturale chiedersi se si possa fare ancora meglio, usando *un solo simbolo*.

Vien naturale chiedersi se si possa fare ancora meglio, usando *un solo simbolo*. Proviamo: prendiamo il simbolo '*' e codifichiamo così:

Vien naturale chiedersi se si possa fare ancora meglio, usando *un solo simbolo*. Proviamo: prendiamo il simbolo '*' e codifichiamo così:

$$0 \longrightarrow * \qquad 1 \longrightarrow * * .$$

Vien naturale chiedersi se si possa fare ancora meglio, usando *un solo simbolo*. Proviamo: prendiamo il simbolo '*' e codifichiamo così:

$$0 \longrightarrow * \qquad 1 \longrightarrow * * .$$

Il nostro messaggio diventerebbe allora

0	0	0	1	0	1	1	0...
*	*	*	**	*	**	**	*...

Vien naturale chiedersi se si possa fare ancora meglio, usando *un solo simbolo*. Proviamo: prendiamo il simbolo '*' e codifichiamo così:

$$0 \longrightarrow * \qquad 1 \longrightarrow **.$$

Il nostro messaggio diventerebbe allora

0	0	0	1	0	1	1	0...
*	*	*	**	*	**	**	*...

cioè

Vien naturale chiedersi se si possa fare ancora meglio, usando *un solo simbolo*. Proviamo: prendiamo il simbolo '*' e codifichiamo così:

$$0 \longrightarrow * \qquad 1 \longrightarrow **.$$

Il nostro messaggio diventerebbe allora

0	0	0	1	0	1	1	0...
*	*	*	**	*	**	**	*...

cioè

$$**********...?????$$

Vien naturale chiedersi se si possa fare ancora meglio, usando *un solo simbolo*. Proviamo: prendiamo il simbolo '*' e codifichiamo così:

$$0 \longrightarrow * \qquad 1 \longrightarrow **.$$

Il nostro messaggio diventerebbe allora

0	0	0	1	0	1	1	0...
*	*	*	**	*	**	**	*...

cioè

$$**********...?????$$

Un momento! Qui non si riconosce più nulla!

Vien naturale chiedersi se si possa fare ancora meglio, usando *un solo simbolo*. Proviamo: prendiamo il simbolo '*' e codifichiamo così:

$$0 \longrightarrow * \qquad 1 \longrightarrow **.$$

Il nostro messaggio diventerebbe allora

0	0	0	1	0	1	1	0...
*	*	*	**	*	**	**	*...

cioè

$$**********...?????$$

Un momento! Qui non si riconosce più nulla! Ora non è più possibile ricostruire nessun messaggio!

Vien naturale chiedersi se si possa fare ancora meglio, usando *un solo simbolo*. Proviamo: prendiamo il simbolo '*' e codifichiamo così:

$$0 \longrightarrow * \qquad 1 \longrightarrow **.$$

Il nostro messaggio diventerebbe allora

0	0	0	1	0	1	1	0...
*	*	*	**	*	**	**	*...

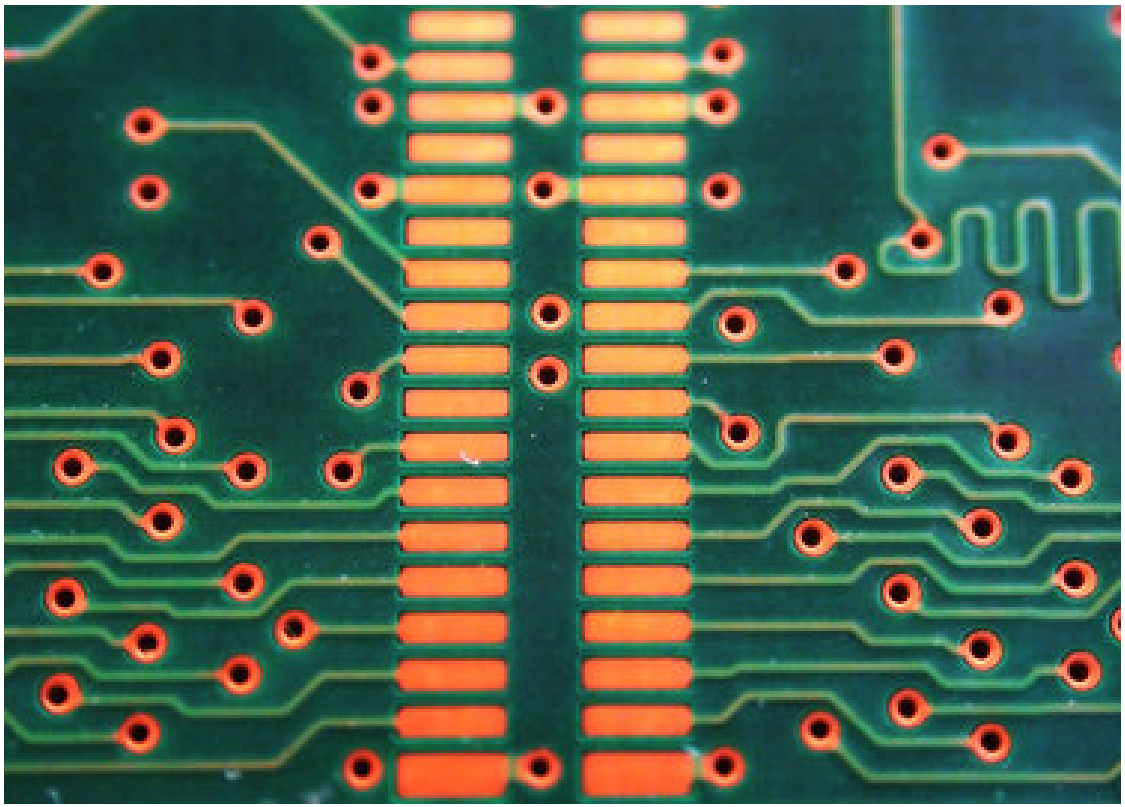
cioè

$$**********...?????$$

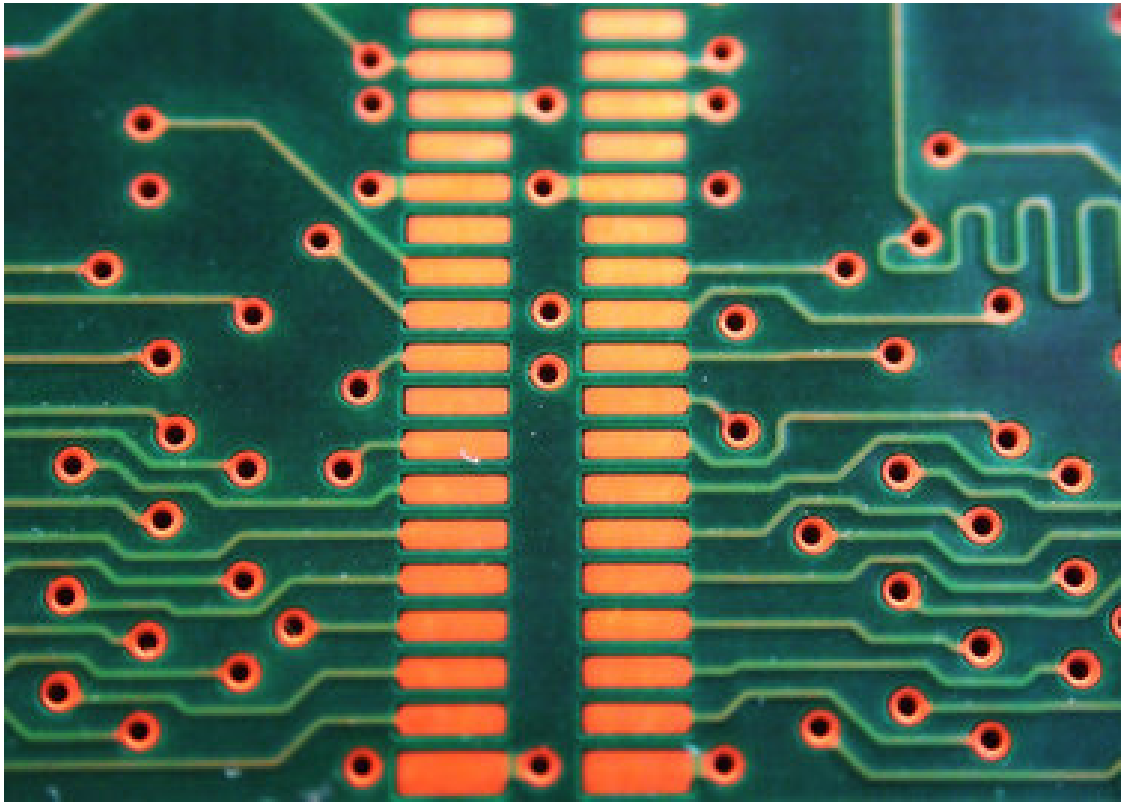
Un momento! Qui non si riconosce più nulla! Ora non è più possibile ricostruire nessun messaggio!Questo dice che non è possibile scendere sotto *due simboli* per trasmettere l'informazione.

Dal punto di vista tecnologico, lo zero e l'uno si realizzano mediante il *passaggio* o meno di corrente attraverso un canale di un microchip...

Dal punto di vista tecnologico, lo zero e l'uno si realizzano mediante il *passaggio* o meno di corrente attraverso un canale di un microchip...



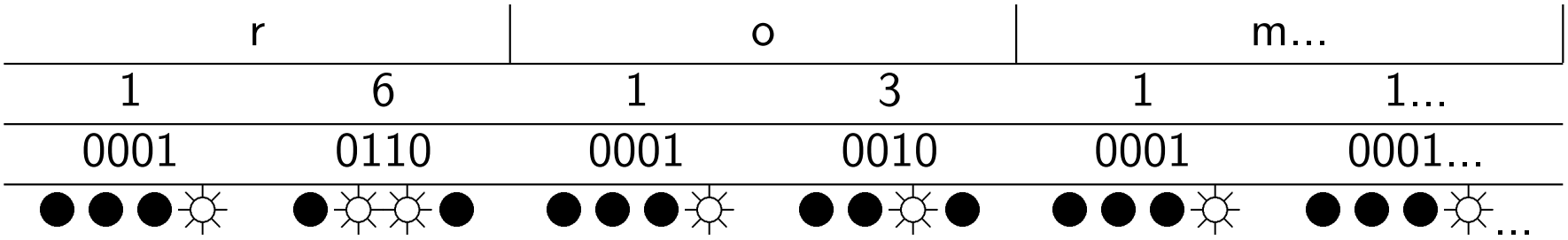
Dal punto di vista tecnologico, lo zero e l'uno si realizzano mediante il *passaggio* o meno di corrente attraverso un canale di un microchip...



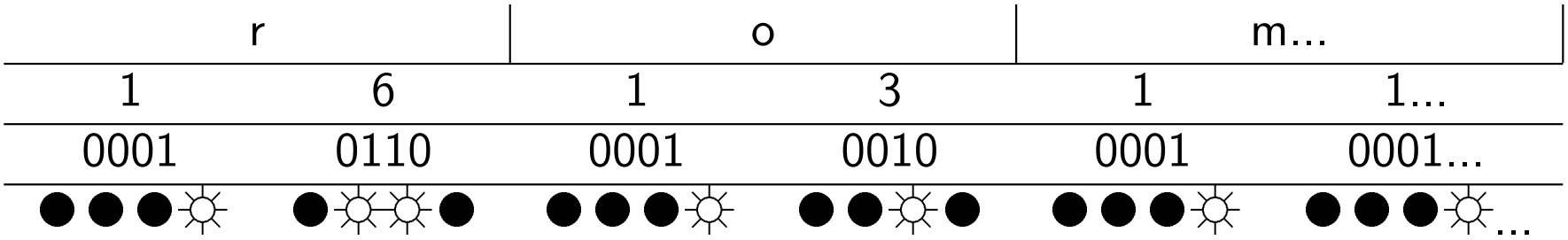
... ma come venga realizzato conta poco; si potrebbe trasmettere informazione anche con altri metodi, ma mai con meno di *due* simboli.

Spesso, in un elaboratore elettronico (computer) gli "uni" e gli "zeri" sono rappresentati con lampadine accese e spente... come qui sotto

Spesso, in un elaboratore elettronico (computer) gli "uni" e gli "zeri" sono rappresentati con lampadine accese e spente... come qui sotto

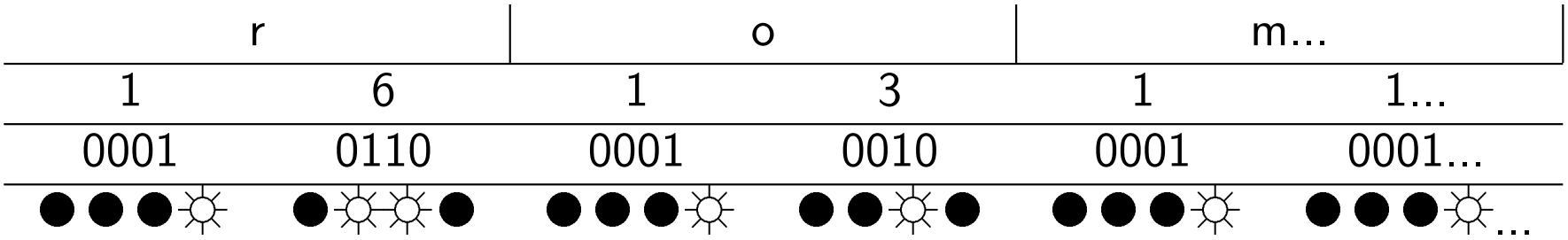


Spesso, in un elaboratore elettronico (computer) gli "uni" e gli "zeri" sono rappresentati con lampadine accese e spente... come qui sotto



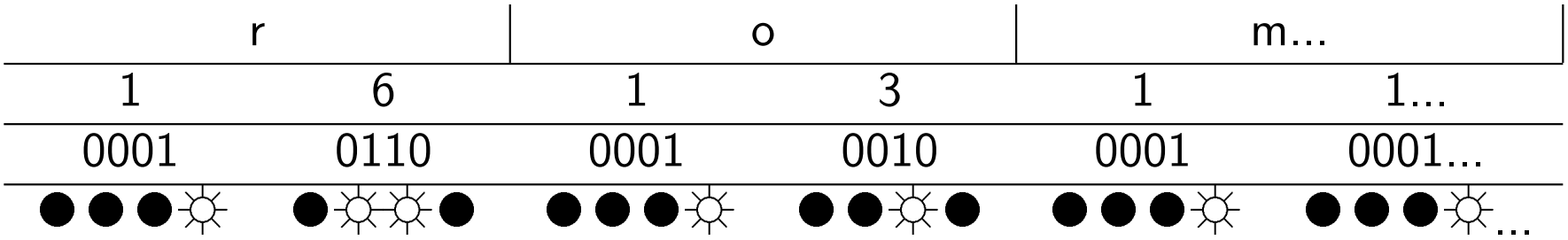
cosicché in pratica si vede questo:

Spesso, in un elaboratore elettronico (computer) gli "uni" e gli "zeri" sono rappresentati con lampadine accese e spente... come qui sotto



cosicché in pratica si vede questo:

Spesso, in un elaboratore elettronico (computer) gli "uni" e gli "zeri" sono rappresentati con lampadine accese e spente... come qui sotto



cosicché in pratica si vede questo:

e in fase di *decodifica* si risale all'informazione come la conosciamo noi.

L'elaborazione

Elaborare un'informazione è una cosa diversa dal rappresentarla.

L'elaborazione

Elaborare un'informazione è una cosa diversa dal rappresentarla.
Per elaborare un'informazione è necessario poter *agire* su di essa.

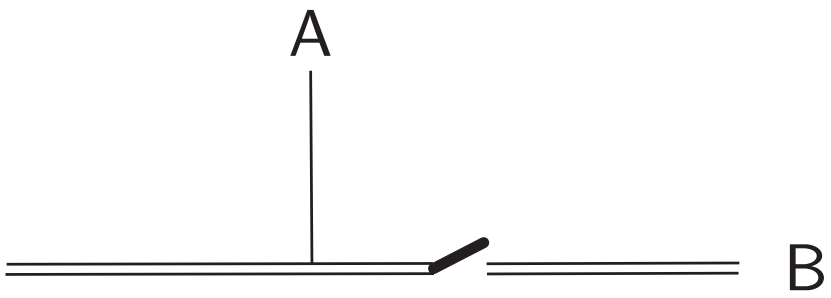
L'elaborazione

Elaborare un'informazione è una cosa diversa dal rappresentarla.
Per elaborare un'informazione è necessario poter *agire* su di essa.
Una rappresentazione comoda delle "lampadine spente e accese" di un computer è quella basata sugli interruttori.

L'elaborazione

Elaborare un'informazione è una cosa diversa dal rappresentarla.
Per elaborare un'informazione è necessario poter *agire* su di essa.
Una rappresentazione comoda delle "lampadine spente e accese" di un computer è quella basata sugli interruttori.

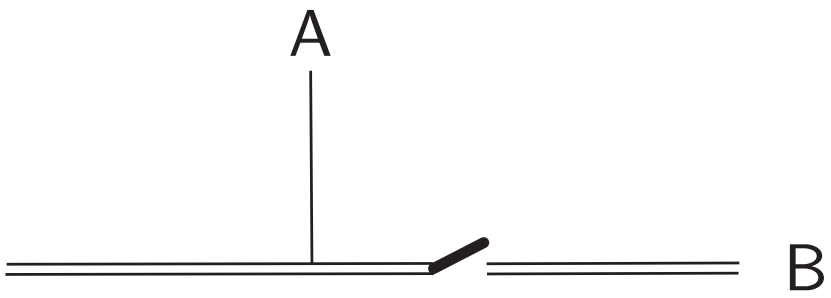
Per essere corretti, però, gli interruttori dei computer sono un po' speciali, sono interruttori "comandati elettricamente":



L'elaborazione

Elaborare un'informazione è una cosa diversa dal rappresentarla.
Per elaborare un'informazione è necessario poter *agire* su di essa.
Una rappresentazione comoda delle "lampadine spente e accese" di un computer è quella basata sugli interruttori.

Per essere corretti, però, gli interruttori dei computer sono un po' speciali, sono interruttori "comandati elettricamente":

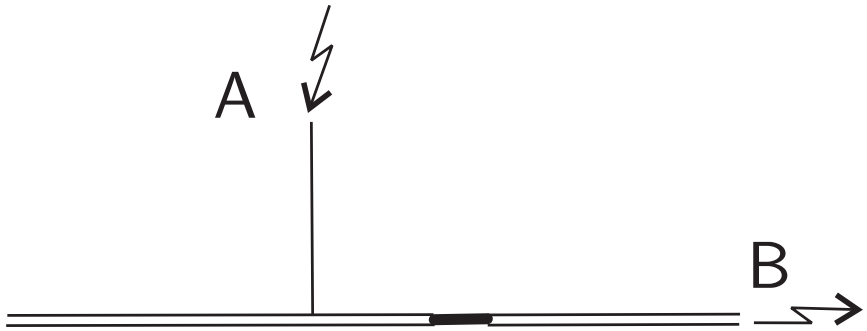


In questo interruttore A rappresenta la linea di comando, che accende/spegne l'interruttore, mentre in B arriva o non arriva corrente.

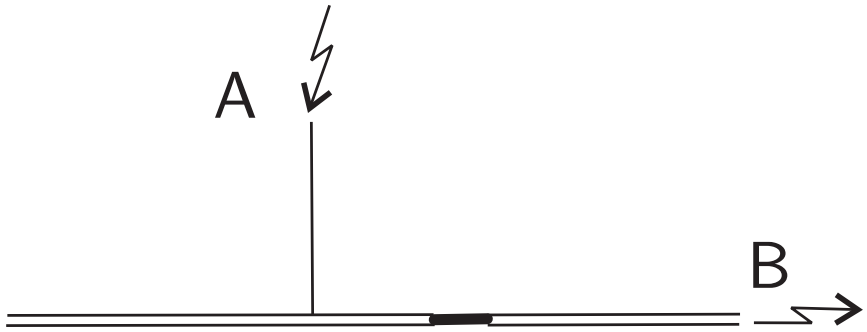
Il nostro interruttore fondamentale funziona così:

Il nostro interruttore fondamentale funziona così: se c'è corrente in A, allora l'interruttore si chiude e c'è corrente anche in B:

Il nostro interruttore fondamentale funziona così: se c'è corrente in A, allora l'interruttore si chiude e c'è corrente anche in B:

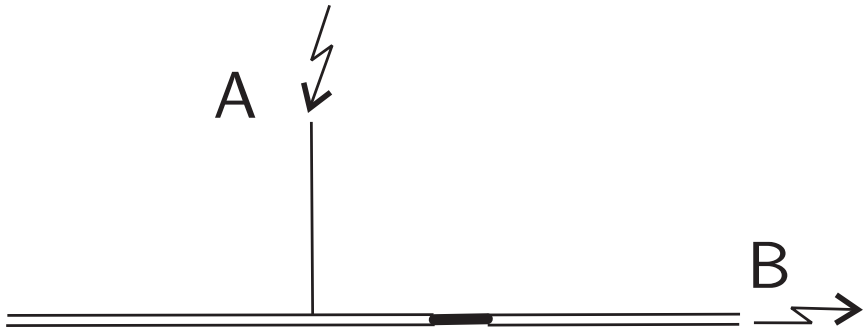


Il nostro interruttore fondamentale funziona così: se c'è corrente in A, allora l'interruttore si chiude e c'è corrente anche in B:

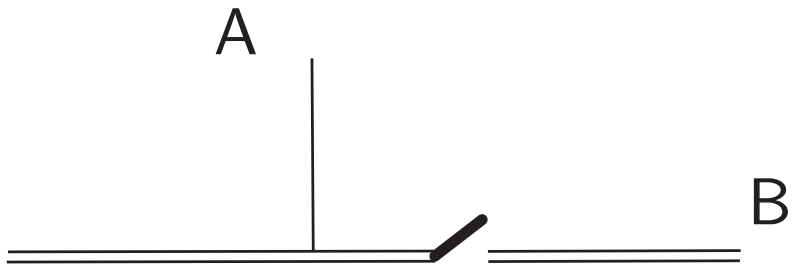


mentre se non c'è corrente in A, allora non c'è neanche in B:

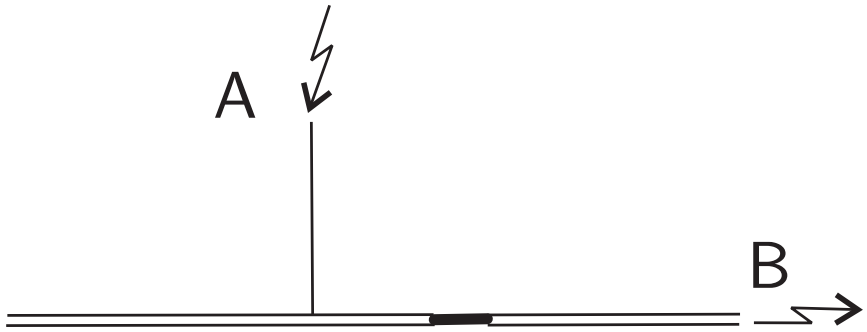
Il nostro interruttore fondamentale funziona così: se c'è corrente in A, allora l'interruttore si chiude e c'è corrente anche in B:



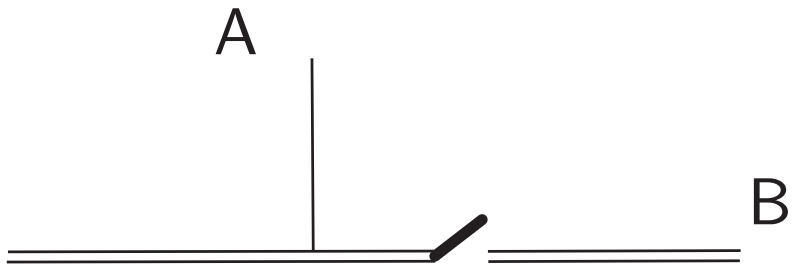
mentre se non c'è corrente in A, allora non c'è neanche in B:



Il nostro interruttore fondamentale funziona così: se c'è corrente in A, allora l'interruttore si chiude e c'è corrente anche in B:



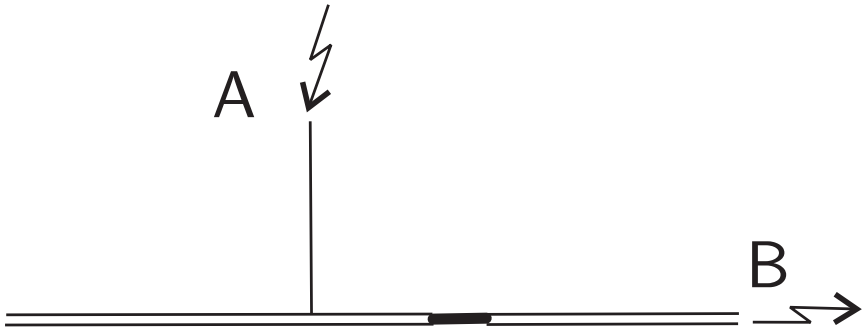
mentre se non c'è corrente in A, allora non c'è neanche in B:



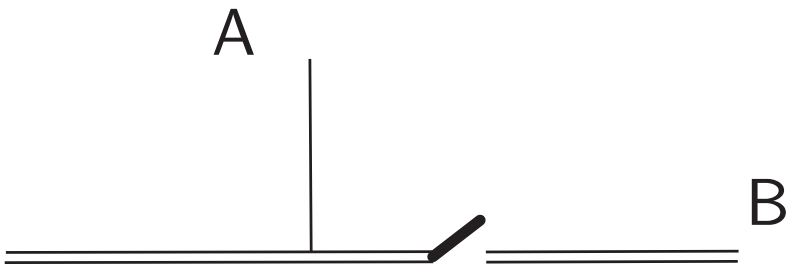
Questo è il funzionamento di un normale *transistor*.

Se rappresentiamo la presenza/assenza di corrente con uno zero (assenza) o con l'uno (presenza), abbiamo uno schema fatto così:

Se rappresentiamo la presenza/assenza di corrente con uno zero (assenza) o con l'uno (presenza), abbiamo uno schema fatto così:



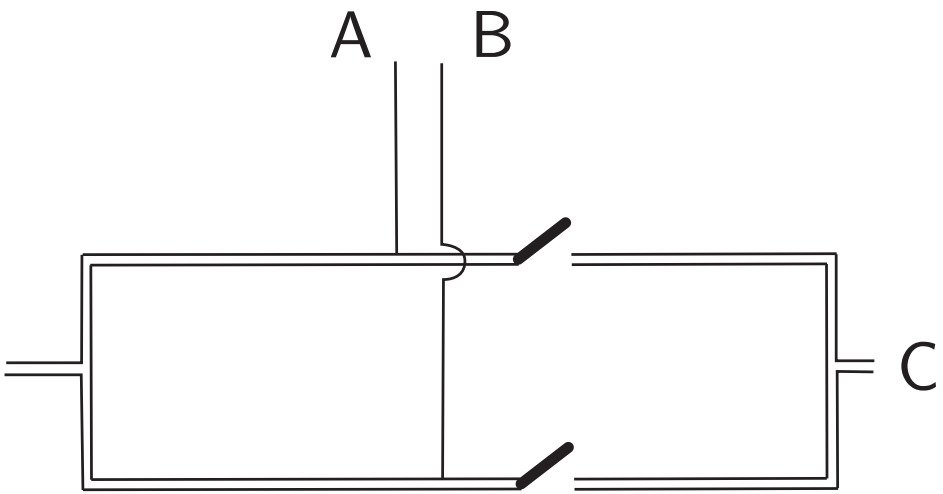
A	B
1	1



A	B
0	0

Vediamo ora cosa succede se colleghiamo *due* di questi interruttori "in parallelo", così:

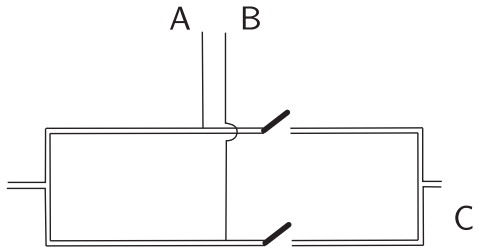
Vediamo ora cosa succede se colleghiamo *due* di questi interruttori "in parallelo", così:



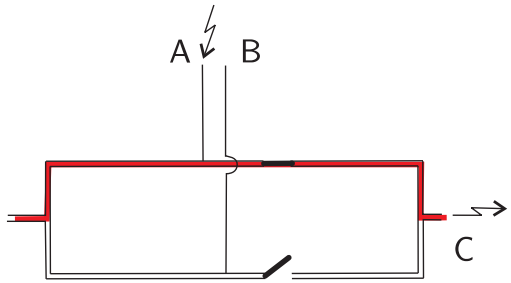
Chiamiamo A e B gli "ingressi" e C il "risultato".

Chiamiamo A e B gli "ingressi" e C il "risultato".
Vediamo i possibili risultati a seconda del passaggio o meno di corrente in A e in B:

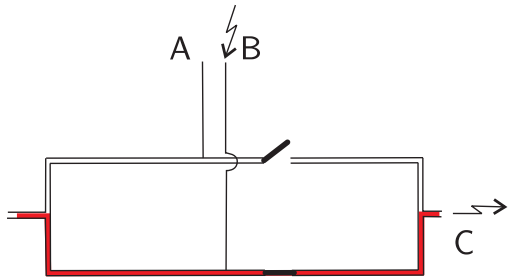
Chiamiamo A e B gli "ingressi" e C il "risultato".
Vediamo i possibili risultati a seconda del passaggio o meno di corrente in A e in B:
(in rosso il percorso della corrente)



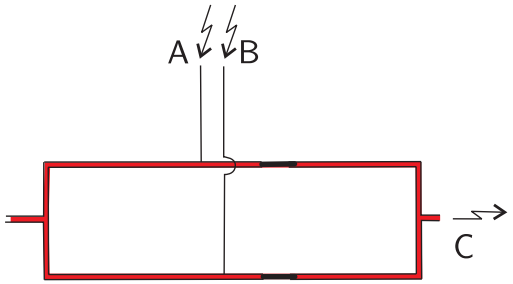
A	B	C
0	0	0



A	B	C
1	0	1



A	B	C
0	1	1



A	B	C
1	1	1

Se riportiamo i valori delle tabelle e accanto la tabella di verità del connettivo "oppure" ($\vee$) che abbiamo studiato in Logica, ci accorgiamo di un fatto straordinario:

Se riportiamo i valori delle tabelle e accanto la tabella di verità del connettivo "oppure" ($\vee$) che abbiamo studiato in Logica, ci accorgiamo di un fatto straordinario:

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

P	Q	P oppure Q
F	F	F
V	F	V
F	V	V
V	V	V

Se riportiamo i valori delle tabelle e accanto la tabella di verità del connettivo "oppure" ($\vee$) che abbiamo studiato in Logica, ci accorgiamo di un fatto straordinario:

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

P	Q	P oppure Q
F	F	F
V	F	V
F	V	V
V	V	V

Le tabelle sono *identiche*, a patto di scrivere "vero" (V) per 1 e "falso" (F) per 0.

Se riportiamo i valori delle tabelle e accanto la tabella di verità del connettivo "oppure" ($\vee$) che abbiamo studiato in Logica, ci accorgiamo di un fatto straordinario:

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

P	Q	P oppure Q
F	F	F
V	F	V
F	V	V
V	V	V

Le tabelle sono *identiche*, a patto di scrivere "vero" (V) per 1 e "falso" (F) per 0. Questa importante corrispondenza si chiama *isomorfismo* e sta alla base del fatto che Matematica e Informatica sono imparentate.

Se riportiamo i valori delle tabelle e accanto la tabella di verità del connettivo "oppure" ($\vee$) che abbiamo studiato in Logica, ci accorgiamo di un fatto straordinario:

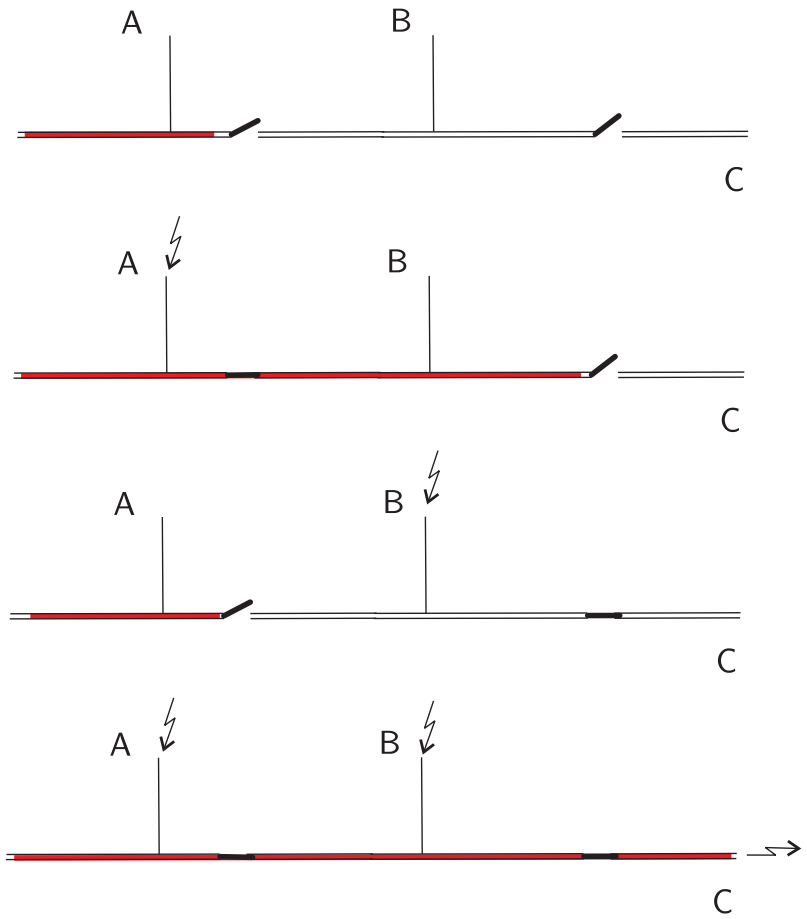
A	B	C
0	0	0
1	0	1
0	1	1
1	1	1

P	Q	P oppure Q
F	F	F
V	F	V
F	V	V
V	V	V

Le tabelle sono *identiche*, a patto di scrivere "vero" (V) per 1 e "falso" (F) per 0. Questa importante corrispondenza si chiama *isomorfismo* e sta alla base del fatto che Matematica e Informatica sono imparentate. Questa "operazione" tra A e B si indica con $\oplus$.

Se invece mettiamo gli interruttori *uno dopo l'altro*, oppure come si dice "in serie", troviamo questo schema di funzionamento:

Se invece mettiamo gli interruttori *uno dopo l'altro*, oppure come si dice "in serie", troviamo questo schema di funzionamento:



A	B	C
0	0	0

A	B	C
1	0	0

A	B	C
0	1	0

A	B	C
1	1	1

Questa tabella è formalmente identica a quella del connettivo "e" ($\wedge$):

Questa tabella è formalmente identica a quella del connettivo "e" ($\wedge$):

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

P	Q	P e Q
F	F	F
V	F	F
F	V	F
V	V	V

Questa tabella è formalmente identica a quella del connettivo "e" ($\wedge$):

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

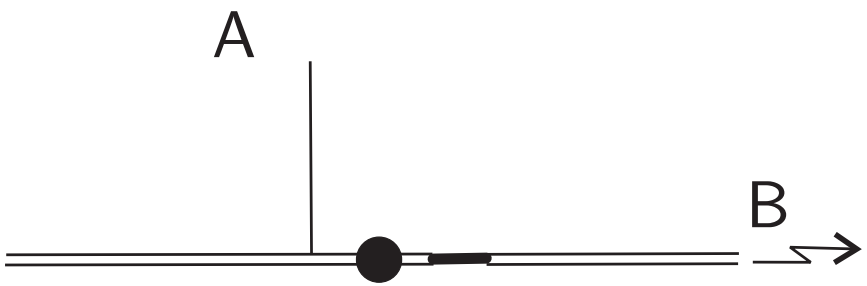
P	Q	P e Q
F	F	F
V	F	F
F	V	F
V	V	V

e l'operazione si indica con $\otimes$.

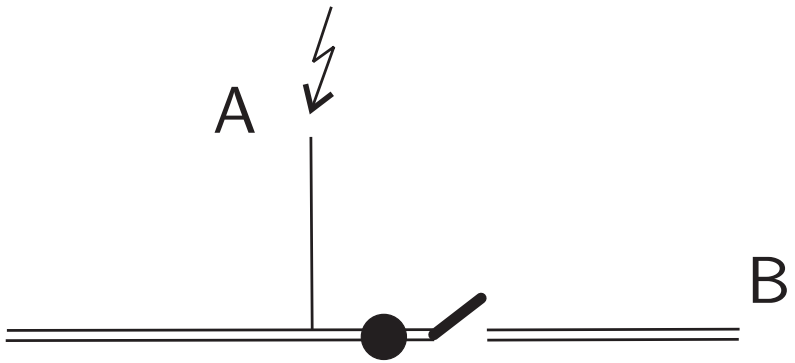
E siccome ricordiamo anche il connettivo "non", che invertiva il valore di verità di una proposizione, ci immaginiamo che esista un interruttore "invertito" rispetto a quello che abbiamo descritto.

E siccome ricordiamo anche il connettivo "non", che invertiva il valore di verità di una proposizione, ci immaginiamo che esista un interruttore "invertito" rispetto a quello che abbiamo descritto.
Esso è descritto come in figura (abbiamo messo il pallino per distinguerlo dall'interruttore "consueto"):

E siccome ricordiamo anche il connettivo "non", che invertiva il valore di verità di una proposizione, ci immaginiamo che esista un interruttore "invertito" rispetto a quello che abbiamo descritto.
Esso è descritto come in figura (abbiamo messo il pallino per distinguerlo dall'interruttore "consueto"):



A	B
0	1



A	B
1	0

In questo modo è possibile *sommare e intersecare logicamente* delle informazioni, in una parola *elaborarle*.

In questo modo è possibile *sommare e intersecare logicamente* delle informazioni, in una parola *elaborarle*.
Nel prossimo capitolo (approfondimento) vedremo una applicazione semplice ma importante: l'addizione.

L'addizione su un bit

Il nostro obiettivo è realizzare l'addizione di due numeri a una cifra (un bit).

L'addizione su un bit

Il nostro obiettivo è realizzare l'addizione di due numeri a una cifra (un bit).
Gli *addendi* sono quindi solo 0 e 1.

L'addizione su un bit

Il nostro obiettivo è realizzare l'addizione di due numeri a una cifra (un bit).
Gli *addendi* sono quindi solo 0 e 1.
Chiaramente, le regole sono le seguenti:

$$0 + 0 = 0$$

L'addizione su un bit

Il nostro obiettivo è realizzare l'addizione di due numeri a una cifra (un bit).
Gli *addendi* sono quindi solo 0 e 1.
Chiaramente, le regole sono le seguenti:

$$0 + 0 = 0$$
$$1 + 0 = 1$$

L'addizione su un bit

Il nostro obiettivo è realizzare l'addizione di due numeri a una cifra (un bit).
Gli *addendi* sono quindi solo 0 e 1.
Chiaramente, le regole sono le seguenti:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$

L'addizione su un bit

Il nostro obiettivo è realizzare l'addizione di due numeri a una cifra (un bit).
Gli *addendi* sono quindi solo 0 e 1.
Chiaramente, le regole sono le seguenti:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$1 + 1 = 10$$

dove chiaramente "10" sta per "2" in binario.

L'addizione su un bit

Il nostro obiettivo è realizzare l'addizione di due numeri a una cifra (un bit).
Gli *addendi* sono quindi solo 0 e 1.
Chiaramente, le regole sono le seguenti:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$1 + 1 = 10$$

dove chiaramente "10" sta per "2" in binario.
Naturalmente abbiamo subito un problema: come rappresentare 10?

L'addizione su un bit

Il nostro obiettivo è realizzare l'addizione di due numeri a una cifra (un bit).
Gli *addendi* sono quindi solo 0 e 1.
Chiaramente, le regole sono le seguenti:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$1 + 1 = 10$$

dove chiaramente "10" sta per "2" in binario.
Naturalmente abbiamo subito un problema: come rappresentare 10?
Accontentiamoci per ora di restituire 0, poi per il "riporto" vedremo...

Abbiamo allora una prima operazione, che indichiamo ancora con $+$, con queste regole:

Abbiamo allora una prima operazione, che indichiamo ancora con $+$, con queste regole:

$$0 + 0 = 0$$

Abbiamo allora una prima operazione, che indichiamo ancora con $+$, con queste regole:

$$0 + 0 = 0$$
$$1 + 0 = 1$$

Abbiamo allora una prima operazione, che indichiamo ancora con $+$, con queste regole:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$

Abbiamo allora una prima operazione, che indichiamo ancora con $+$, con queste regole:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$1 + 1 = 0.$$

Abbiamo allora una prima operazione, che indichiamo ancora con $+$, con queste regole:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$1 + 1 = 0.$$

Ci chiediamo come realizzare questo con i nostri circuiti.

Possiamo allora avvalerci dell'isomorfismo scoperto prima. Ricordiamo dalla Logica che l'espressione

Possiamo allora avvalerci dell'isomorfismo scoperto prima. Ricordiamo dalla Logica che l'espressione

$$[P \text{ e } (\text{non } Q)] \text{ oppure } [(\text{non } P) \text{ e } Q]$$

Possiamo allora avvalerci dell'isomorfismo scoperto prima. Ricordiamo dalla Logica che l'espressione

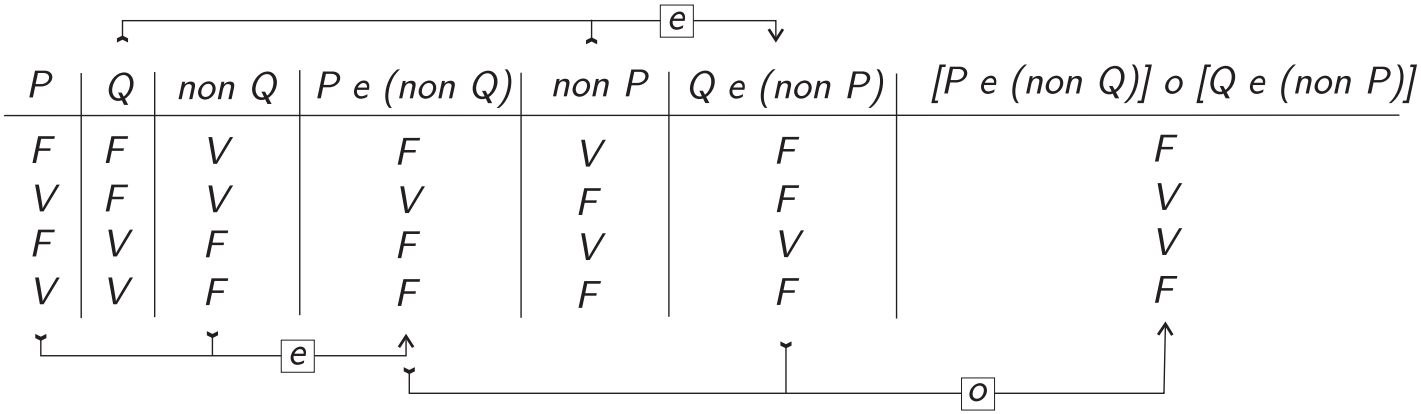
$$[P \text{ e } (\text{non } Q)] \text{ oppure } [(\text{non } P) \text{ e } Q]$$

ha proprio i valori di verità che vogliamo (F V V F, che nel nostro isomorfismo corrispondono a 0 1 1 0), e infatti

Possiamo allora avvalerci dell'isomorfismo scoperto prima. Ricordiamo dalla Logica che l'espressione

$$[P \text{ e } (\text{non } Q)] \text{ oppure } [(\text{non } P) \text{ e } Q]$$

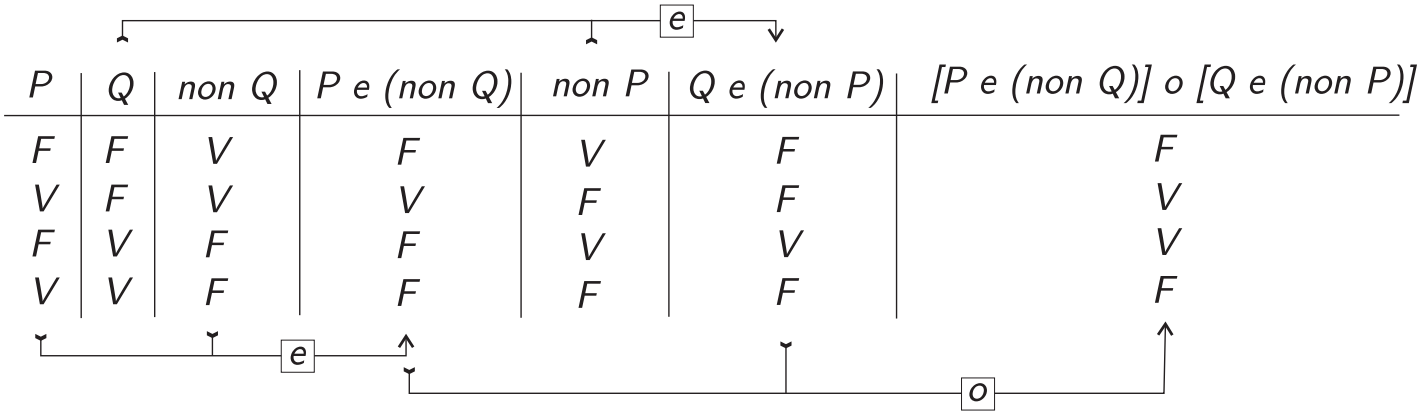
ha proprio i valori di verità che vogliamo (F V V F, che nel nostro isomorfismo corrispondono a 0 1 1 0), e infatti



Possiamo allora avvalerci dell'isomorfismo scoperto prima. Ricordiamo dalla Logica che l'espressione

$$[P \text{ e } (\text{non } Q)] \text{ oppure } [(\text{non } P) \text{ e } Q]$$

ha proprio i valori di verità che vogliamo (F V V F, che nel nostro isomorfismo corrispondono a 0 1 1 0), e infatti

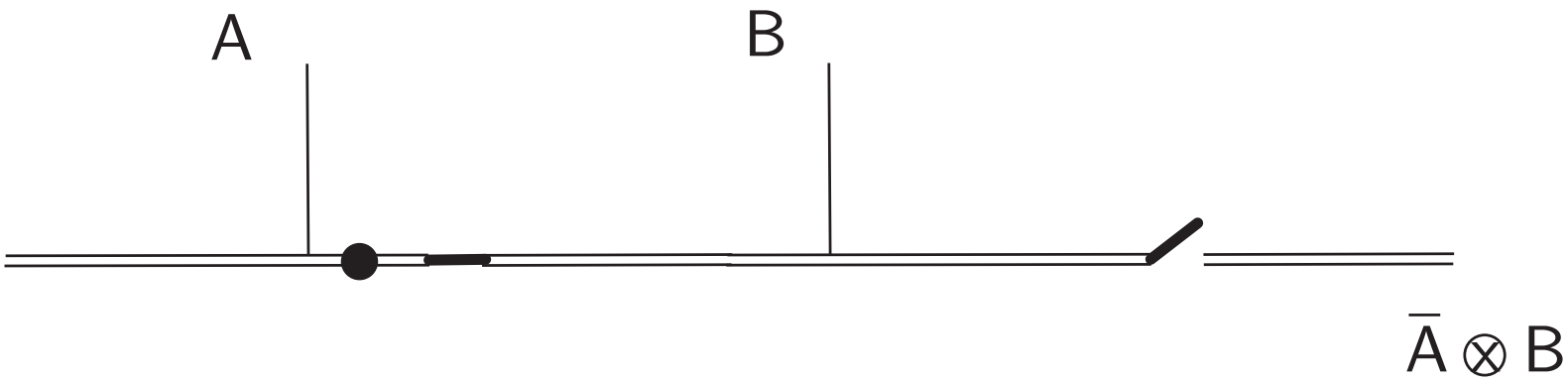


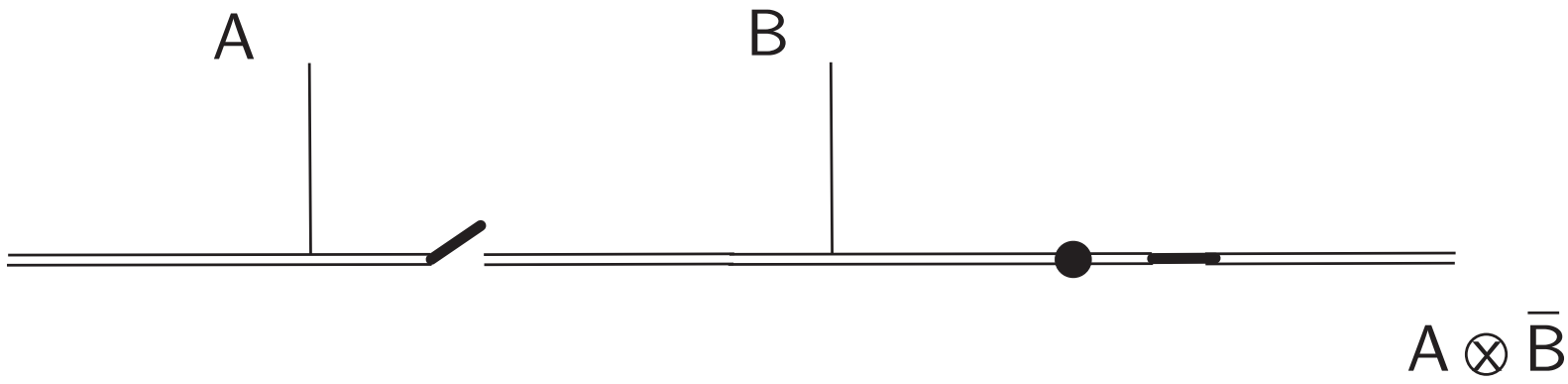
Allora, grazie all'isomorfismo, avremo che

$$A + B = (\underbrace{A}_{A} \underbrace{\otimes}_{\text{e}} \underbrace{\overline{B}}_{\text{non } B}) \underbrace{\oplus}_{\text{o}} (\underbrace{\overline{A}}_{\text{non } A} \underbrace{\otimes}_{\text{e}} \underbrace{B}_{B}).$$

Allora proviamo a costruire $A \otimes \overline{B}$ e $\overline{A} \otimes B$ mettendo in serie due interruttori, uno normale e uno invertito:

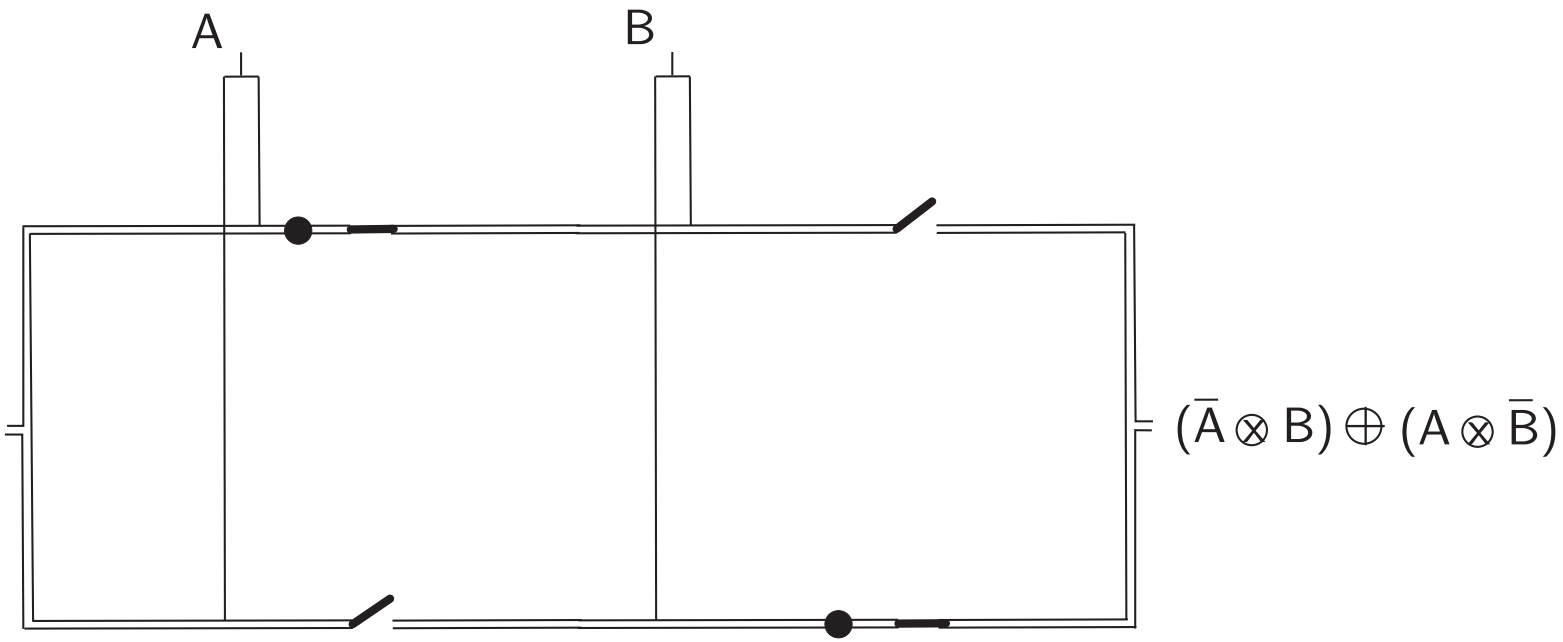
Allora proviamo a costruire $A \otimes \overline{B}$ e $\overline{A} \otimes B$ mettendo in serie due interruttori, uno normale e uno invertito:





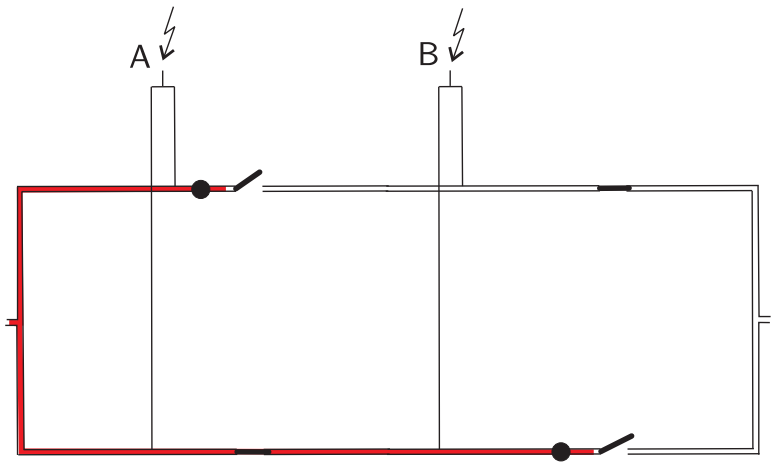
Per effettuare la somma logica dei due pezzi appena costruiti, bisogna "duplicare" la linea di ingresso di A e B (perché compaiono *due* volte nell'espressione $(A \otimes \overline{B}) \oplus (\overline{A} \otimes B)$):

Per effettuare la somma logica dei due pezzi appena costruiti, bisogna "duplicare" la linea di ingresso di A e B (perché compaiono *due* volte nell'espressione $(A \otimes \overline{B}) \oplus (\overline{A} \otimes B)$):

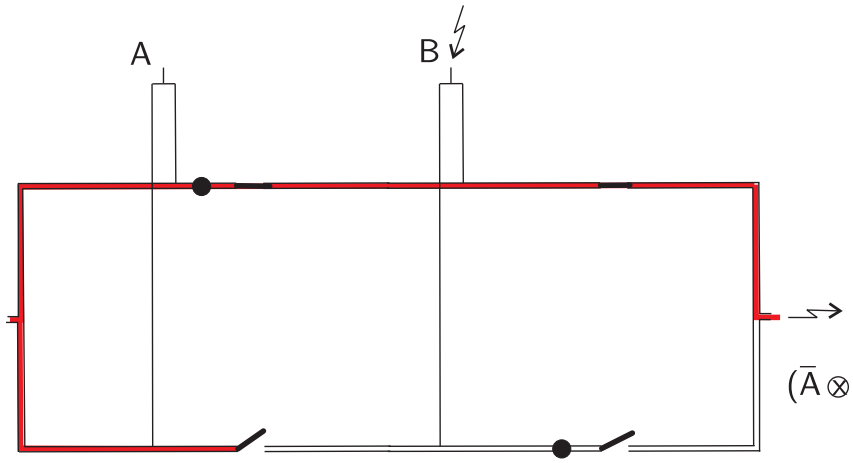


Possiamo verificare che questo circuito funziona secondo le nostre aspettative. Qui sotto sono illustrati due casi:

Possiamo verificare che questo circuito funziona secondo le nostre aspettative. Qui sotto sono illustrati due casi:



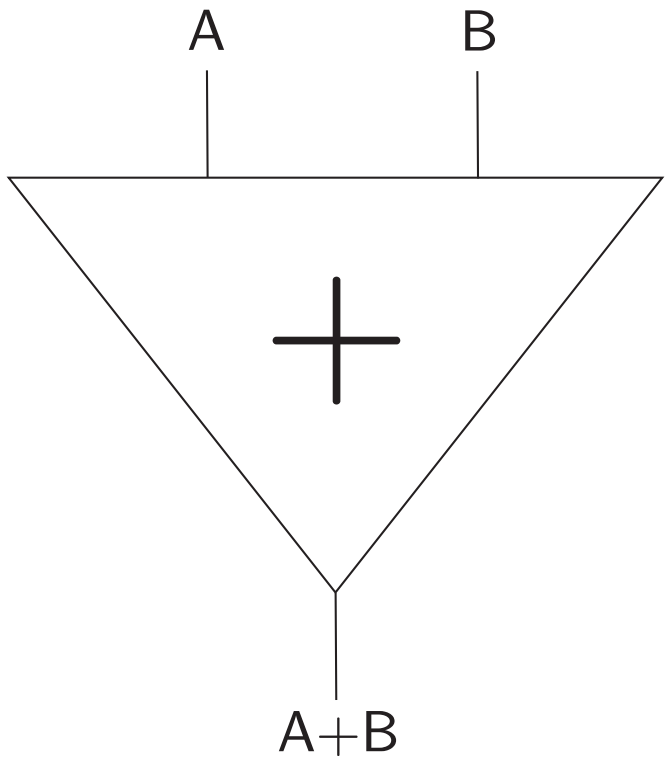
$(\bar{A} \otimes B) \oplus (A \otimes \bar{B})$



$(\bar{A} \otimes B) \oplus (A \otimes \bar{B})$

Possiamo inventare un simbolo abbreviato per la nostra macchinetta: essa ha due linee in ingresso e una in uscita; la possiamo indicare così:

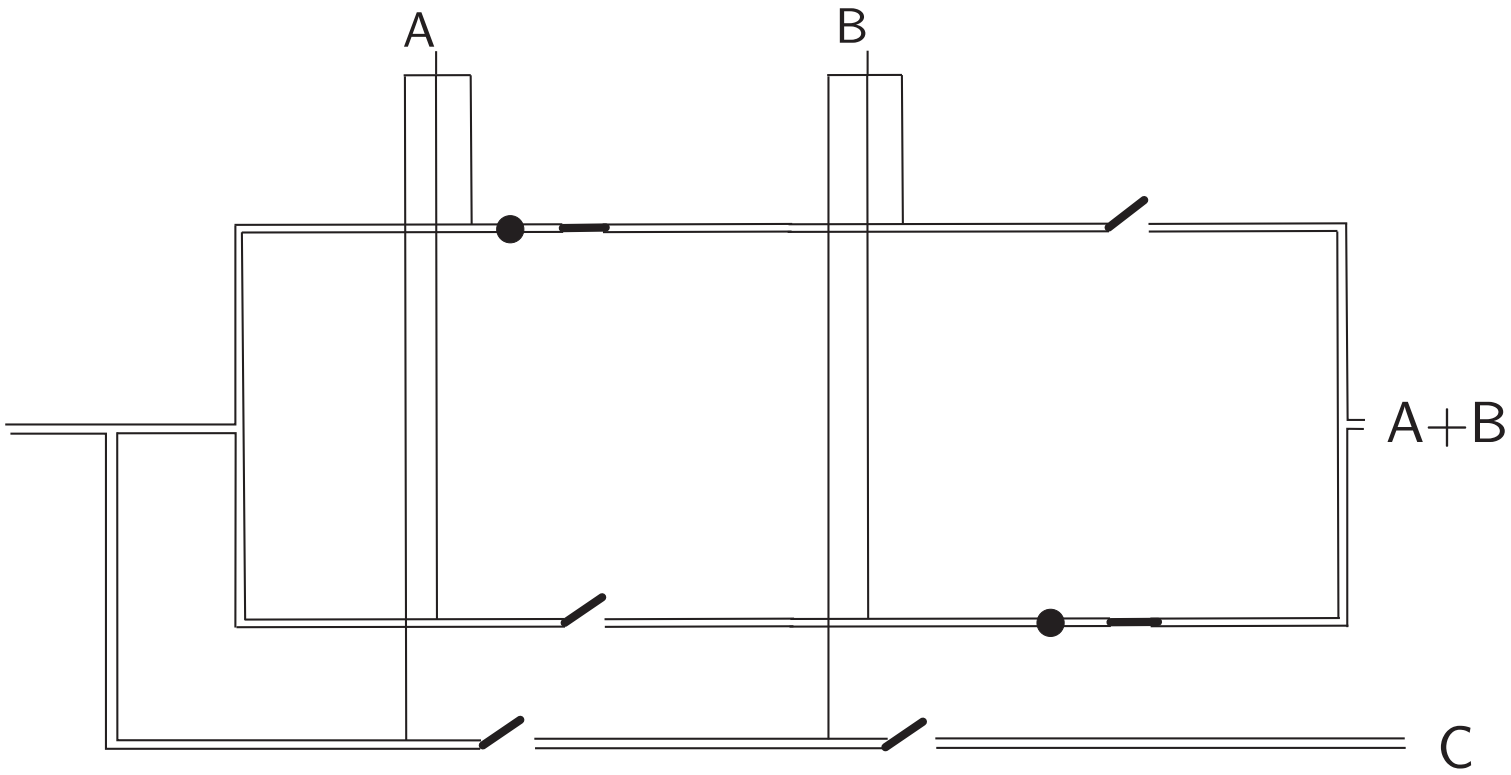
Possiamo inventare un simbolo abbreviato per la nostra macchinetta: essa ha due linee in ingresso e una in uscita; la possiamo indicare così:



E il riporto?

E il riporto? Quello è facile da ottenere: è 1 solo se *sia A che B* sono 1, per cui basta fare il prodotto logico di A e B (naturalmente serve un'altra derivazione, e una anche per la corrente in ingresso):

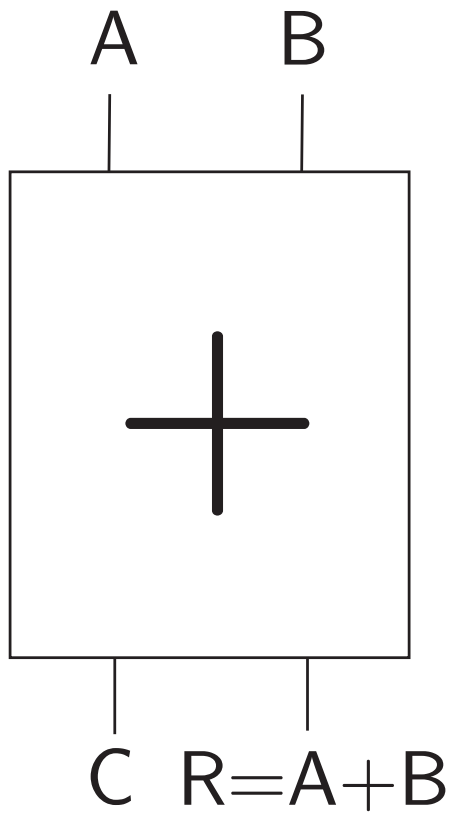
E il riporto? Quello è facile da ottenere: è 1 solo se *sia A che B* sono 1, per cui basta fare il prodotto logico di A e B (naturalmente serve un'altra derivazione, e una anche per la corrente in ingresso):



Questa apparecchiatura è un *sommatore su un bit*. In ingresso vi sono i due bit da sommare (A e B) e in uscita vi è la somma $R = A + B$ (il risultato) e il riporto C (dall'inglese *carry*).

Questa apparecchiatura è un *sommatore su un bit*. In ingresso vi sono i due bit da sommare (A e B) e in uscita vi è la somma $R = A + B$ (il risultato) e il riporto C (dall'inglese *carry*).
Possiamo schematizzarla così:

Questa apparecchiatura è un *sommatore su un bit*. In ingresso vi sono i due bit da sommare (A e B) e in uscita vi è la somma $R = A + B$ (il risultato) e il riporto C (dall'inglese *carry*).
Possiamo schematizzarla così:



Somme su più bit

L'ultima questione riguarda la possibilità di disporre più sommatori per effettuare somme complesse.

Somme su più bit

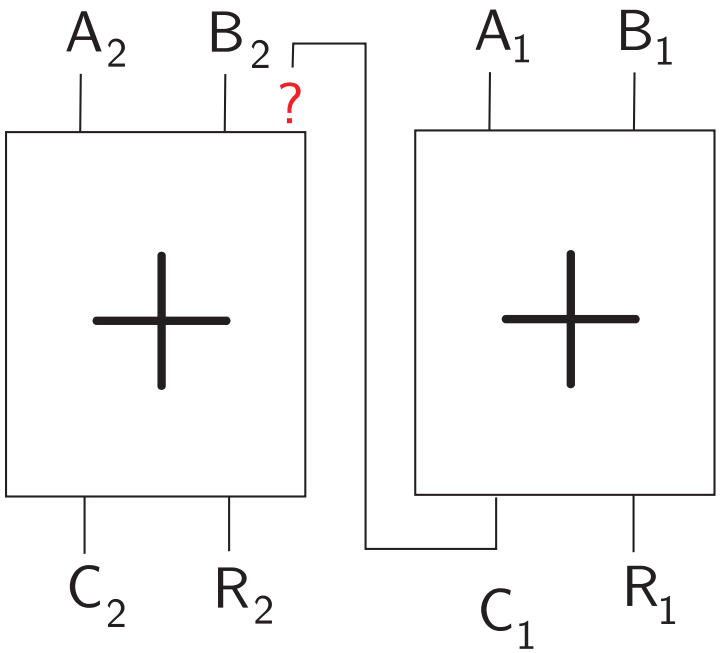
L'ultima questione riguarda la possibilità di disporre più sommatori per effettuare somme complesse. Noi siamo soliti usare il riporto sommato sulla cifra a sinistra come *ingresso* sulle cifre già esistenti.

Somme su più bit

L'ultima questione riguarda la possibilità di disporre più sommatori per effettuare somme complesse. Noi siamo soliti usare il riporto sommato sulla cifra a sinistra come *ingresso* sulle cifre già esistenti.
Però, contando anche il riporto, gli ingressi diventerebbero *tre*, e il nostro sommatore ne ha solo *due*:

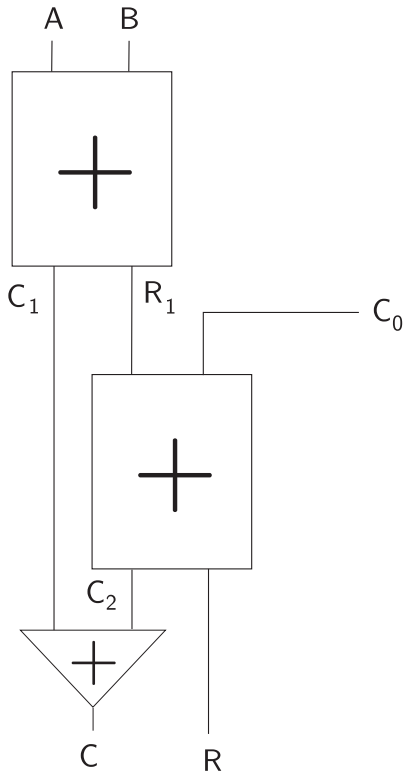
Somme su più bit

L'ultima questione riguarda la possibilità di disporre più sommatori per effettuare somme complesse. Noi siamo soliti usare il riporto sommato sulla cifra a sinistra come *ingresso* sulle cifre già esistenti.
Però, contando anche il riporto, gli ingressi diventerebbero *tre*, e il nostro sommatore ne ha solo *due*:

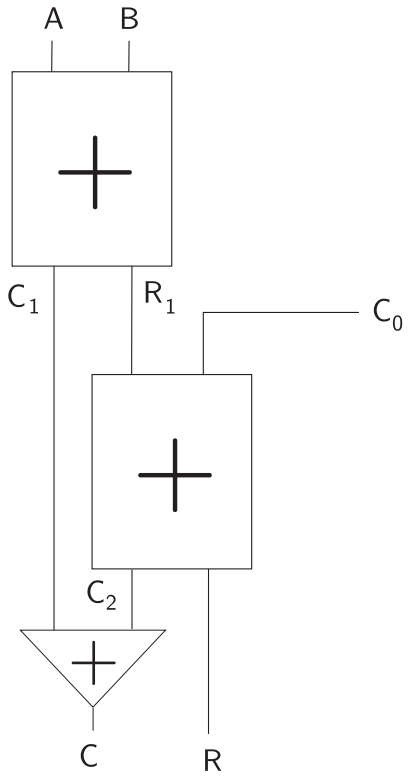


Come possiamo fare? Una soluzione è indicata nella figura qui sotto, un po' tecnica se volete:

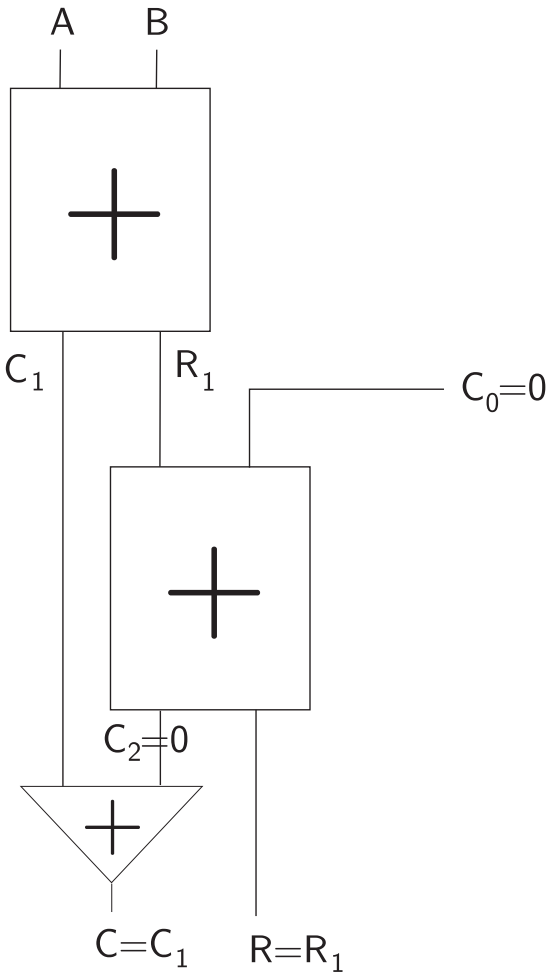
Come possiamo fare? Una soluzione è indicata nella figura qui sotto, un po' tecnica se volete:



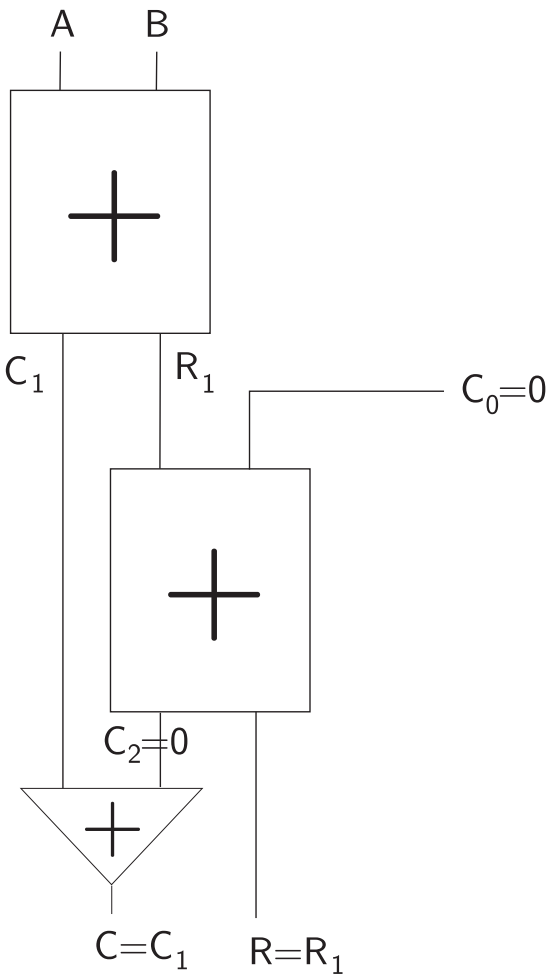
Come possiamo fare? Una soluzione è indicata nella figura qui sotto, un po' tecnica se volete:



La spiegazione è questa: C_0 è il riporto proveniente dall'operazione sulla cifra precedente, mentre A e B sono le nuove cifre da sommare. Alla fine i risultati sono C e R, il nuovo riporto e il nuovo risultato.

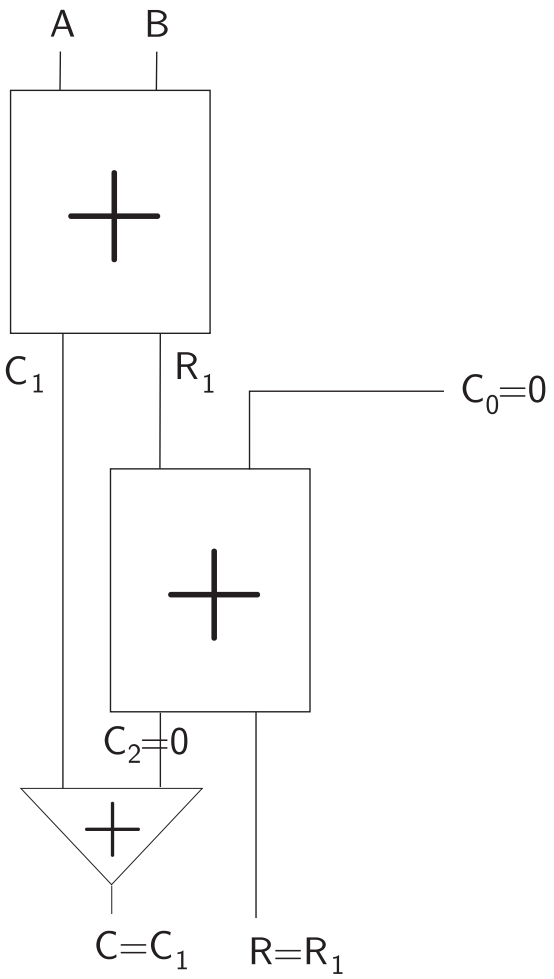


Vediamo perché funziona.



Vediamo perché funziona.

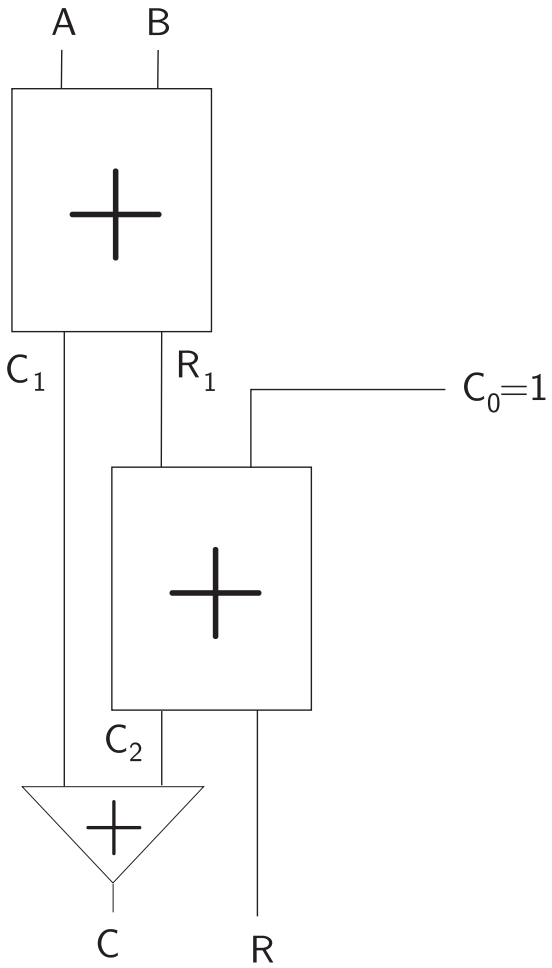
Se il riporto C_0 è zero, il primo risultato intermedio R_1 non cambia, perché sommando con zero non cambia nulla.



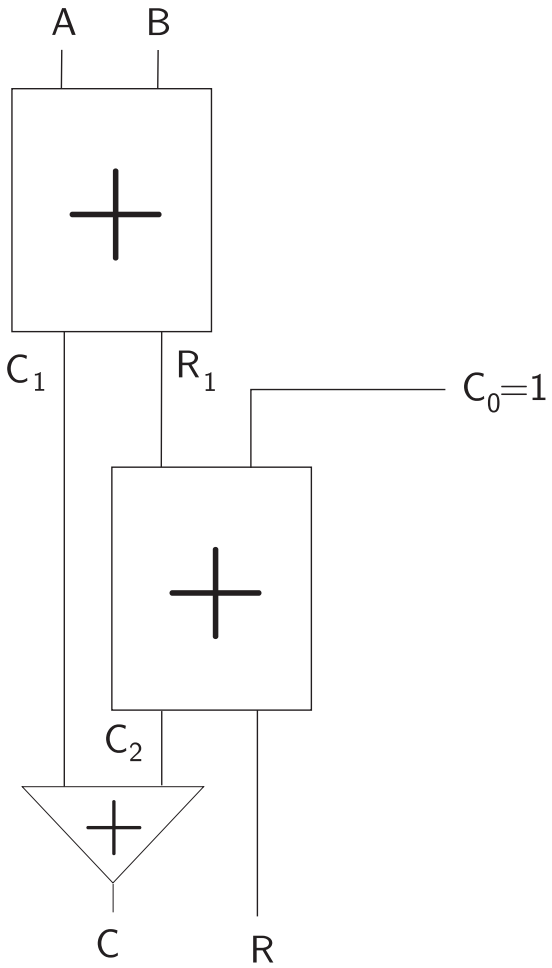
Vediamo perché funziona.

Se il riporto C_0 è zero, il primo risultato intermedio R_1 non cambia, perché sommando con zero non cambia nulla.
Inoltre, sommando zero non può venire $C_2 = 1$, per cui C_1 resta uguale e alla fine

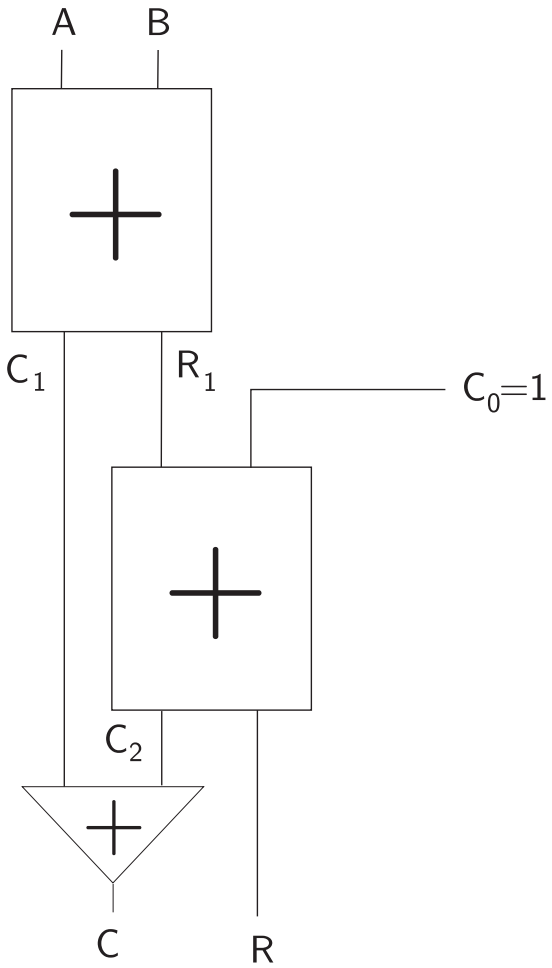
$$R = A + B, \quad C = C_1.$$



Se il riporto C_0 è uno, allora le cose cambiano.

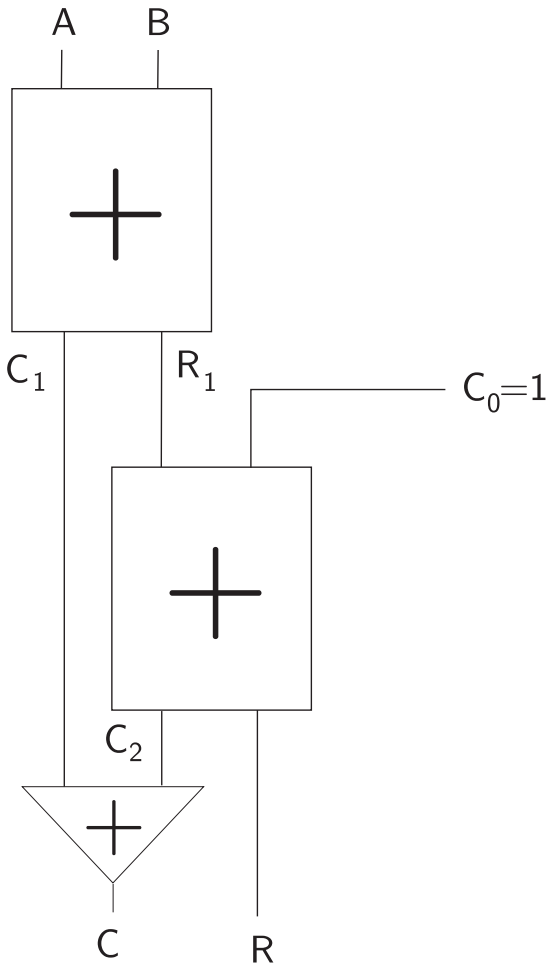


Se il riporto C_0 è uno, allora le cose cambiano.
A questo punto abbiamo tre casi:



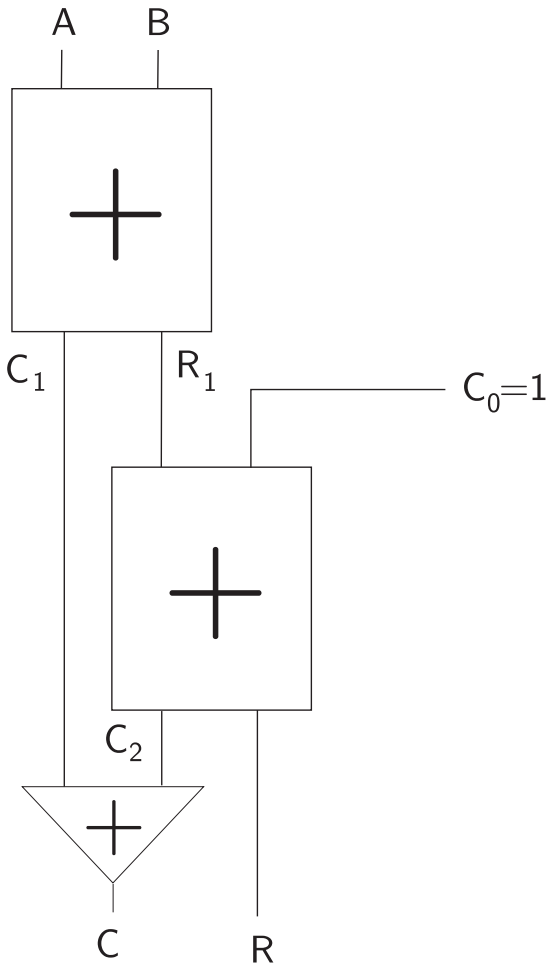
Se il riporto C_0 è uno, allora le cose cambiano. A questo punto abbiamo tre casi:

- A e B entrambi nulli;



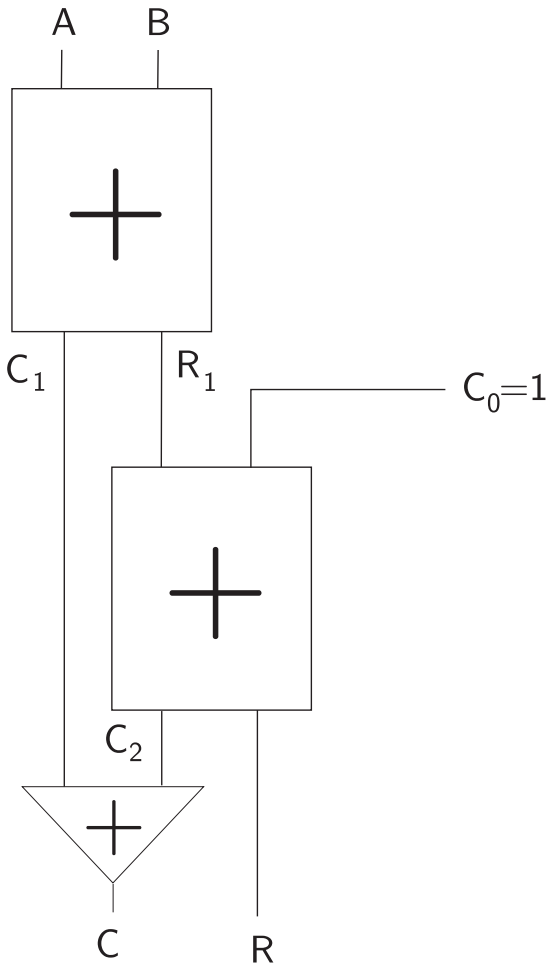
Se il riporto C_0 è uno, allora le cose cambiano. A questo punto abbiamo tre casi:

- A e B entrambi nulli;
- A e B diversi;



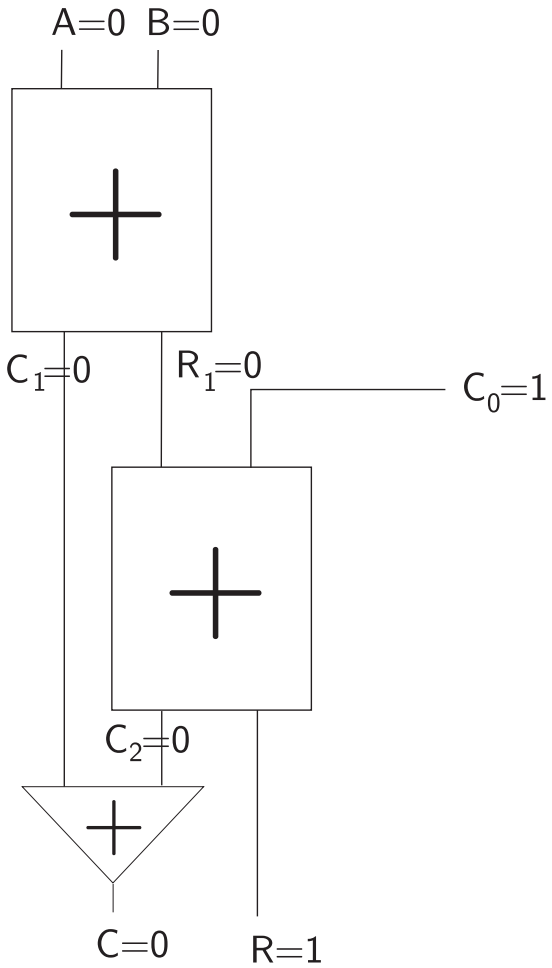
Se il riporto C_0 è uno, allora le cose cambiano. A questo punto abbiamo tre casi:

- A e B entrambi nulli;
- A e B diversi;
- A e B entrambi 1

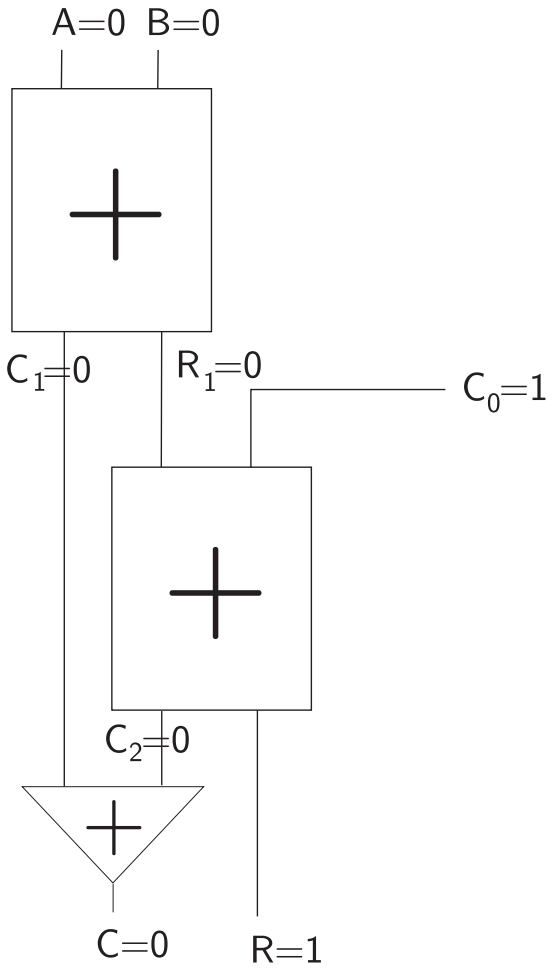


Se il riporto C_0 è uno, allora le cose cambiano. A questo punto abbiamo tre casi:

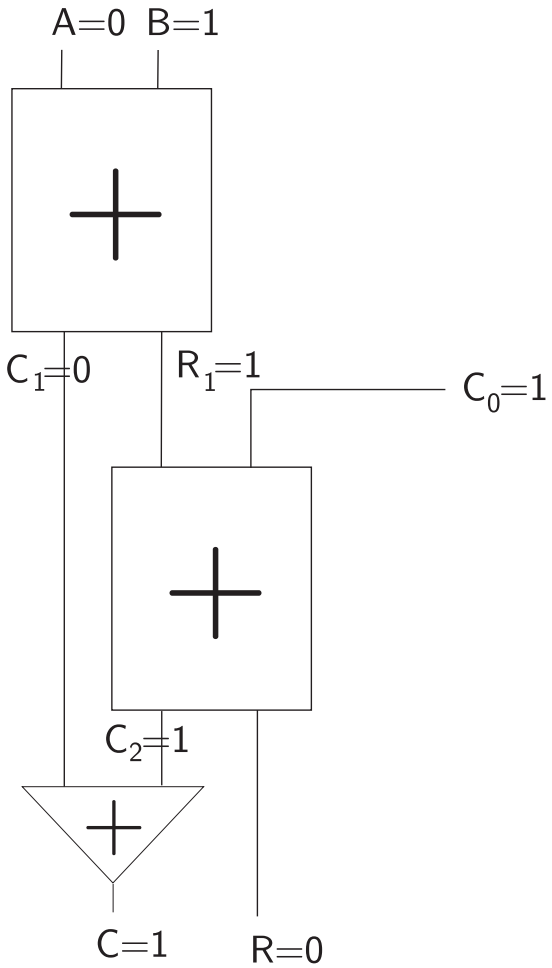
- A e B entrambi nulli;
- A e B diversi;
- A e B entrambi 1



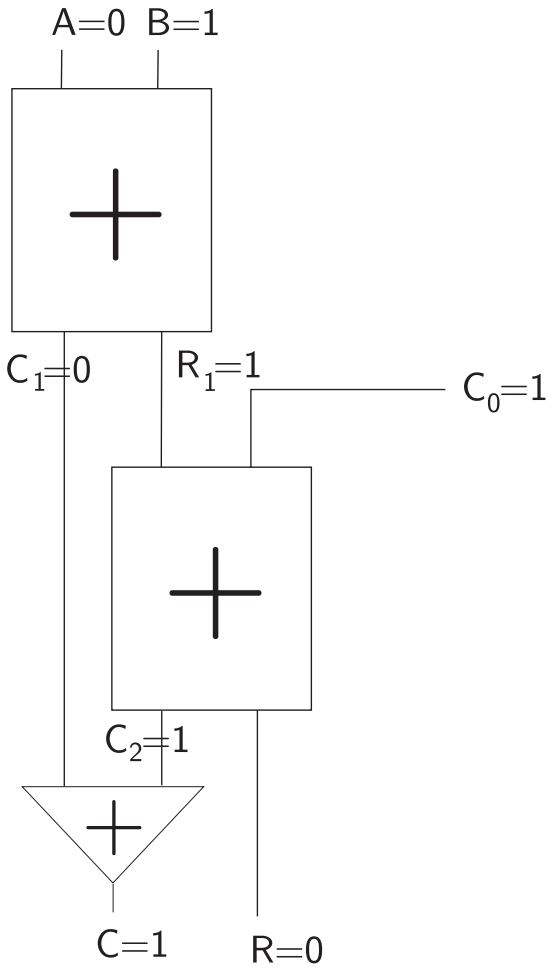
I caso. A e B uguali a zero.



I caso. A e B uguali a zero.
Dunque R_1 è zero e pertanto C_2 non può essere 1, quindi $C = C_1$. Ma allora il risultato è 1 col riporto di zero, come deve essere.

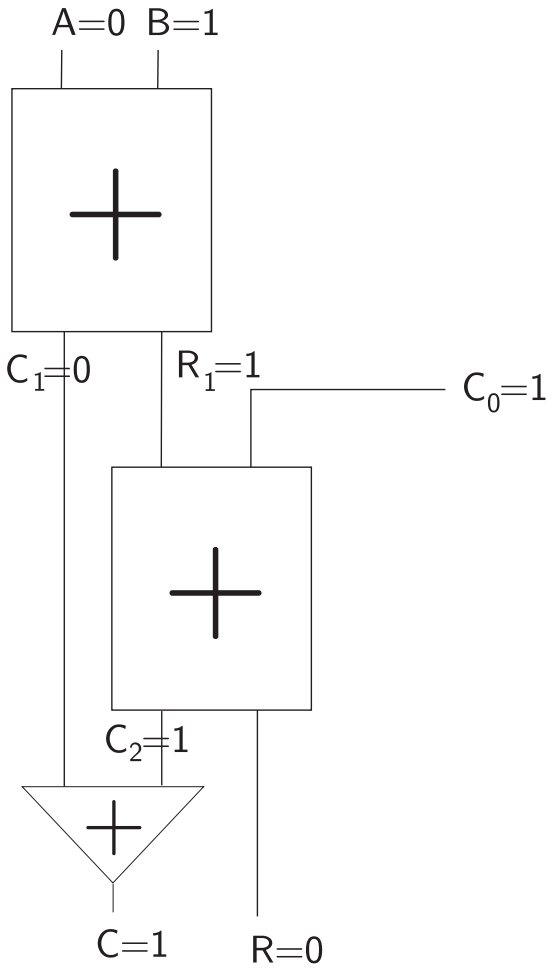


II caso. A e B diversi.



II caso. A e B diversi.
In questo caso C_1 è zero e R_1 è 1, per cui il secondo sommatore dà 0 col riporto C_2 pari a 1. Questo riporto va sommato a C_1, che è nullo, e dà per risultato 1. In definitiva $R = 0$ e $C = 1$, e infatti

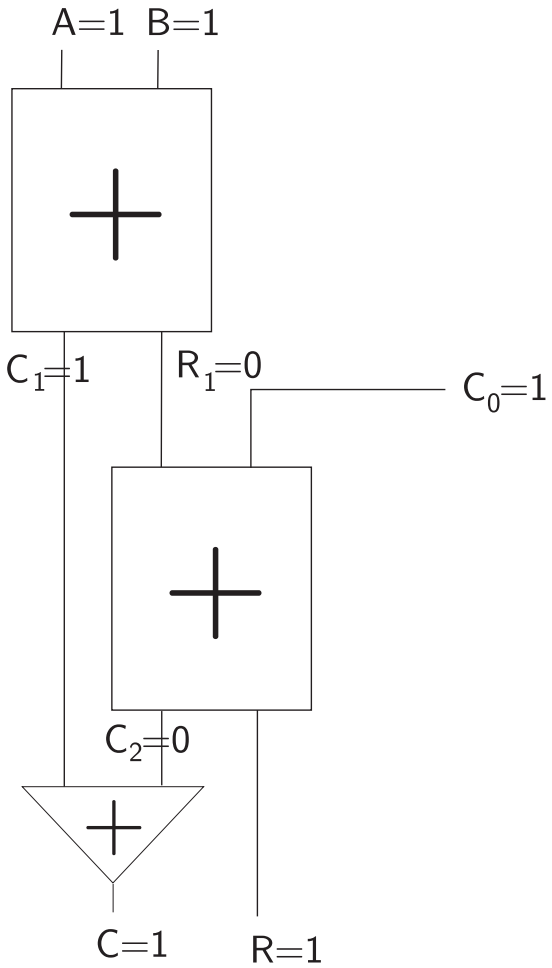
$$\underbrace{1 + 0}_{A+B} + \underbrace{1}_{C_0} = 0 \text{ col riporto di } 1$$



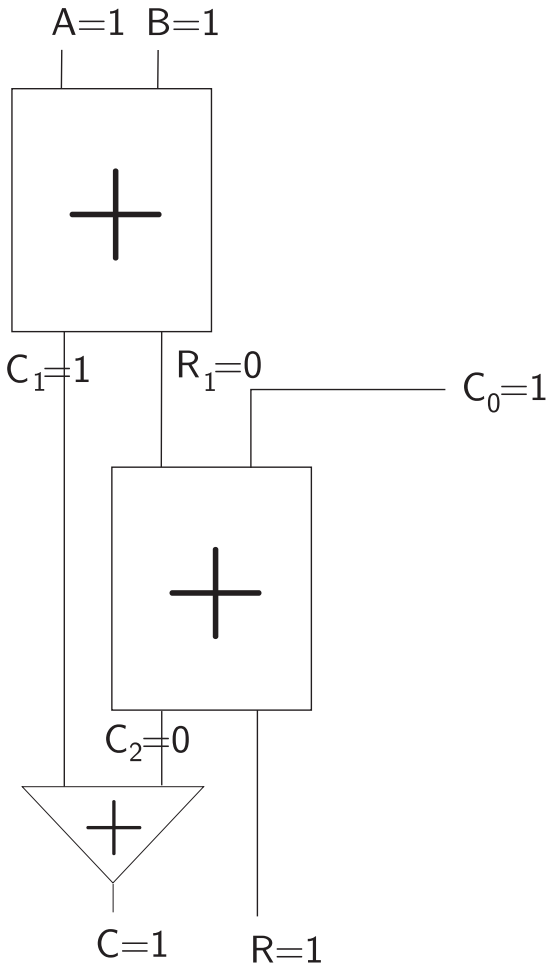
II caso. A e B diversi.
In questo caso C_1 è zero e R_1 è 1, per cui il secondo sommatore dà 0 col riporto C_2 pari a 1. Questo riporto va sommato a C_1, che è nullo, e dà per risultato 1. In definitiva $R = 0$ e $C = 1$, e infatti

$$\underbrace{1+0}_{A+B} + \underbrace{1}_{C_0} = 0 \text{ col riporto di } 1$$

e analogamente se A e B sono scambiati.



III caso. A e B uguali a 1.



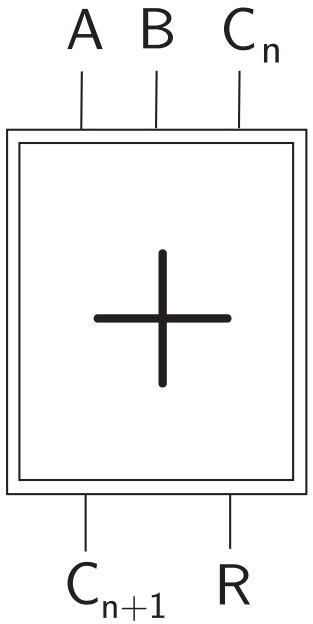
III caso. A e B uguali a 1.
In questo caso C_1 è 1 e R_1 è zero, per cui il secondo sommatore dà 1 col riporto C_2 pari a 0. Questo riporto va sommato a C_1, che è 1, e dà per risultato 1. In definitiva $R = 1$ e C_1, e infatti

$$\underbrace{1+1}_{A+B} + \underbrace{1}_{C_0} = 1 \text{ col riporto di } 1.$$

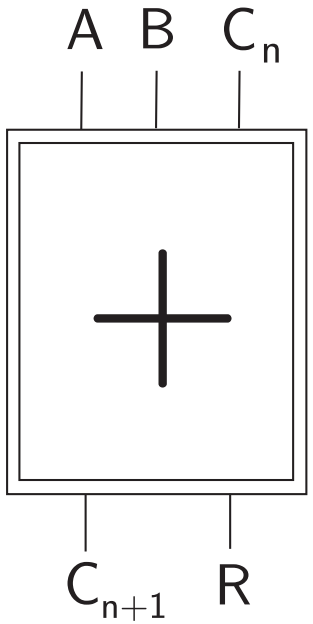
Il sistema funziona, se guardiamo bene, perché in nessun caso da un sommatore su un bit escono due 1, e la somma semplice usa proprio questo fatto, dato che da lì non uscirà mai un riporto.

Il sistema funziona, se guardiamo bene, perché in nessun caso da un sommatore su un bit escono due 1, e la somma semplice usa proprio questo fatto, dato che da lì non uscirà mai un riporto.
Questo è un *sommatore composto*, e possiamo indicarlo così:

Il sistema funziona, se guardiamo bene, perché in nessun caso da un sommatore su un bit escono due 1, e la somma semplice usa proprio questo fatto, dato che da lì non uscirà mai un riporto.
Questo è un *sommatore composto*, e possiamo indicarlo così:



Il sistema funziona, se guardiamo bene, perché in nessun caso da un sommatore su un bit escono due 1, e la somma semplice usa proprio questo fatto, dato che da lì non uscirà mai un riporto.
Questo è un *sommatore composto*, e possiamo indicarlo così:



Come ingresso esso ha le cifre da sommare (A e B) e il riporto della n-esima cifra, mentre in uscita dà il risultato e l'$n+1$-esimo riporto.

Supponiamo infine di avere due numeri "grandi" da sommare, come per esempio $A = 12$ e $B = 9$.

Supponiamo infine di avere due numeri "grandi" da sommare, come per esempio $A = 12$ e $B = 9$. In binario essi sono

Supponiamo infine di avere due numeri "grandi" da sommare, come per esempio $A = 12$ e $B = 9$. In binario essi sono

$$A = 1100 \qquad \text{e} \qquad B = 1001.$$

Supponiamo infine di avere due numeri "grandi" da sommare, come per esempio $A = 12$ e $B = 9$. In binario essi sono

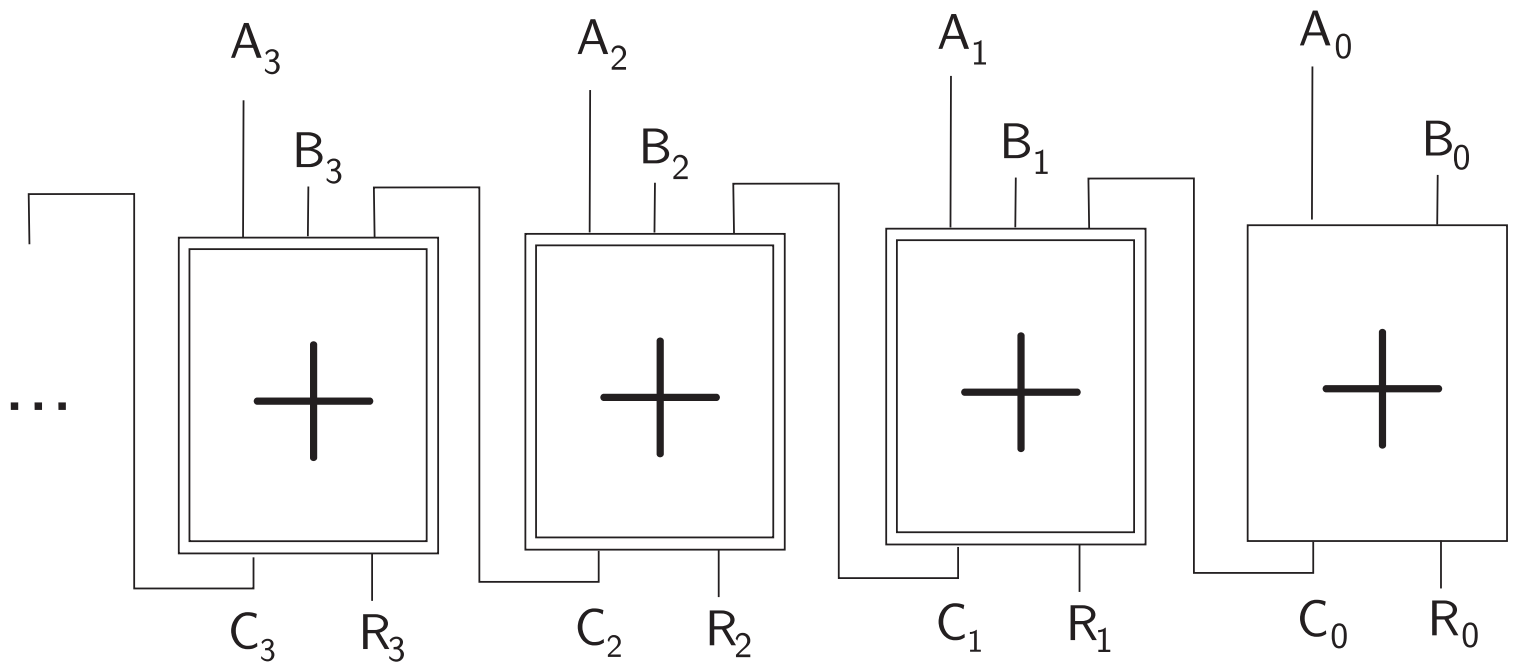
$$A = 1100 \qquad \text{e} \qquad B = 1001.$$

Servono allora quattro bit, perché ognuno dei numeri ha 4 cifre (se erano piccoli si aggiungevano degli zeri). Occorre allora una macchinetta con quattro sommatori composti, così:

Supponiamo infine di avere due numeri "grandi" da sommare, come per esempio $A = 12$ e $B = 9$. In binario essi sono

$$A = 1100 \qquad \text{e} \qquad B = 1001.$$

Servono allora quattro bit, perché ognuno dei numeri ha 4 cifre (se erano piccoli si aggiungevano degli zeri). Occorre allora una macchinetta con quattro sommatori composti, così:



(notate che il primo sommatore—quello più a destra—è "semplice" perché non gli arriva riporto)

Poi decomponiamo i due numeri così:

Poi decomponiamo i due numeri così:

$$\underbrace{1}_{A_0}\ \underbrace{1}_{A_1}\ \underbrace{0}_{A_2}\ \underbrace{0}_{A_3} \quad \text{e} \quad \underbrace{1}_{B_0}\ \underbrace{0}_{B_1}\ \underbrace{0}_{B_2}\ \underbrace{1}_{B_3}$$

Poi decomponiamo i due numeri così:

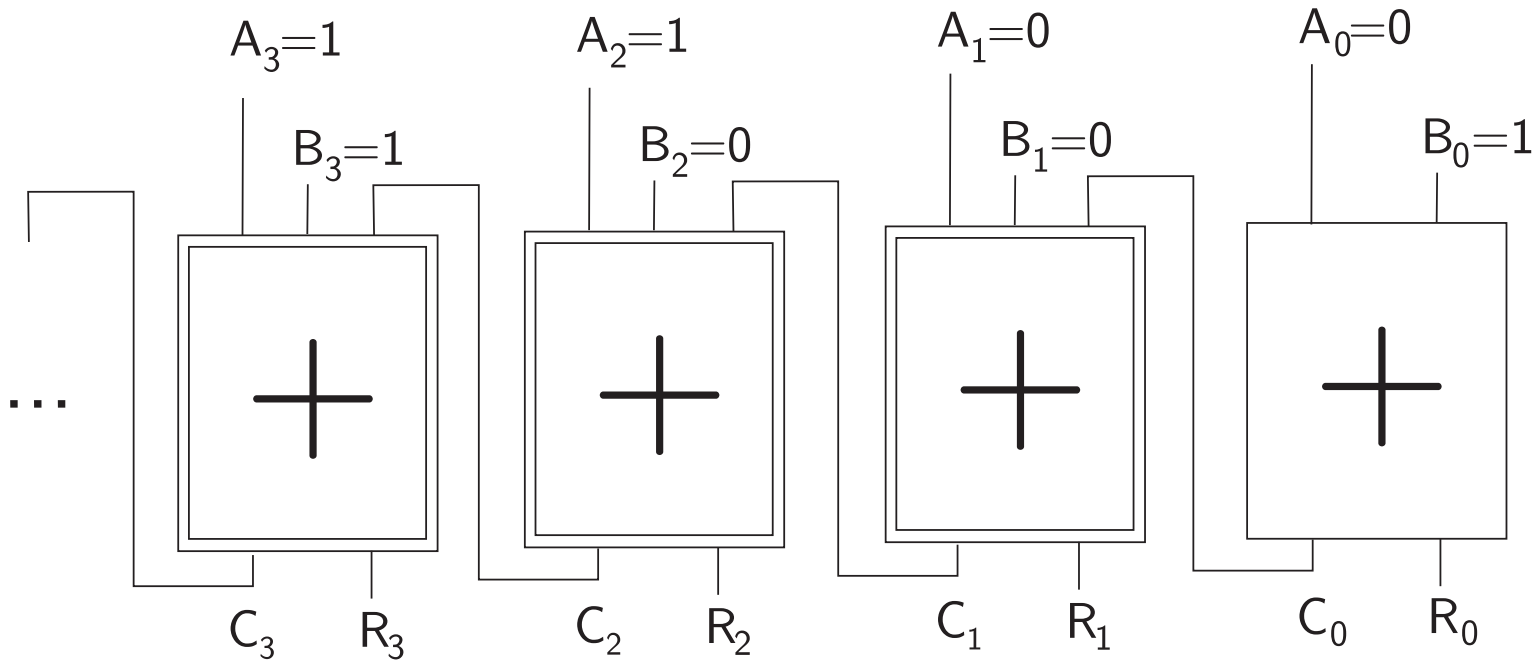
$$\underbrace{1}_{A_0}\ \underbrace{1}_{A_1}\ \underbrace{0}_{A_2}\ \underbrace{0}_{A_3} \quad \text{e} \quad \underbrace{1}_{B_0}\ \underbrace{0}_{B_1}\ \underbrace{0}_{B_2}\ \underbrace{1}_{B_3}$$

e inseriamoli come ingressi nei quattro sommatori:

Poi decomponiamo i due numeri così:

$$\underbrace{1}_{A_0}\ \underbrace{1}_{A_1}\ \underbrace{0}_{A_2}\ \underbrace{0}_{A_3} \quad \text{e} \quad \underbrace{1}_{B_0}\ \underbrace{0}_{B_1}\ \underbrace{0}_{B_2}\ \underbrace{1}_{B_3}$$

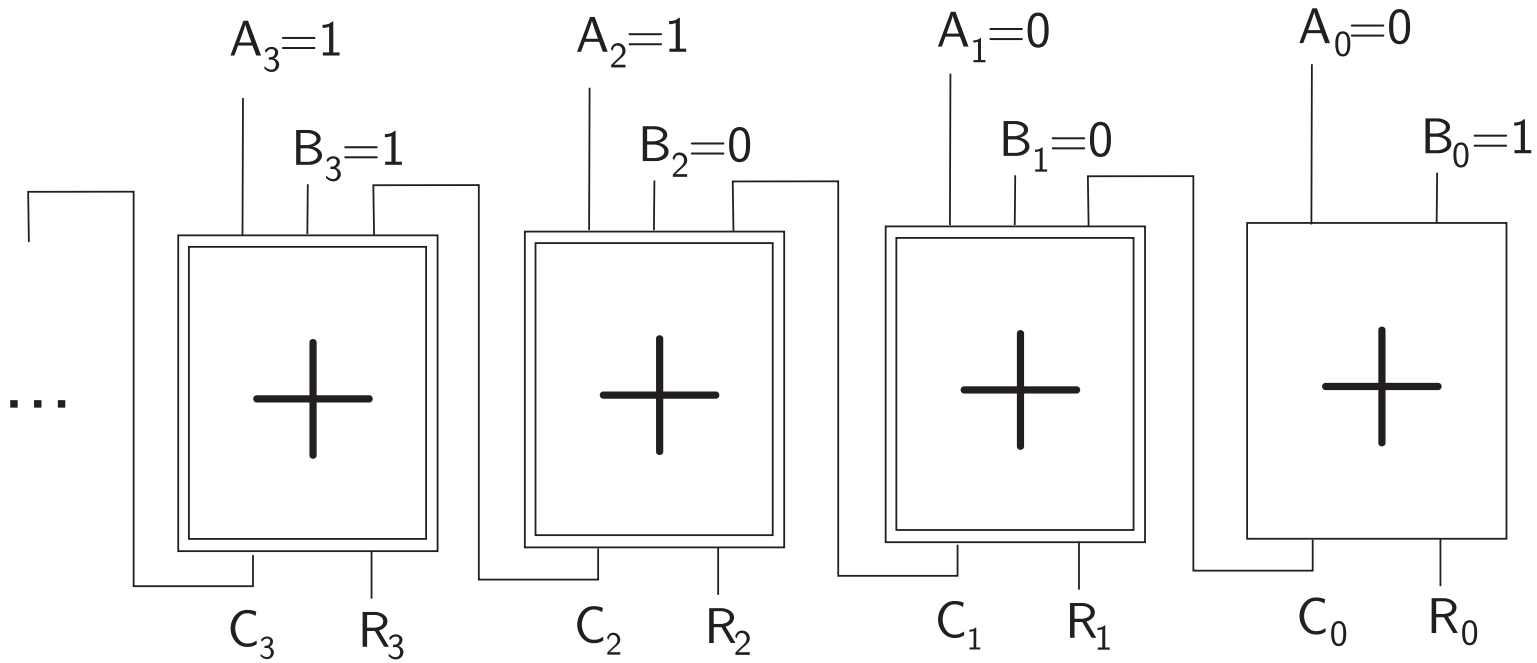
e inseriamoli come ingressi nei quattro sommatori:



Poi decomponiamo i due numeri così:

$$\underbrace{1}_{A_0}\ \underbrace{1}_{A_1}\ \underbrace{0}_{A_2}\ \underbrace{0}_{A_3} \quad \text{e} \quad \underbrace{1}_{B_0}\ \underbrace{0}_{B_1}\ \underbrace{0}_{B_2}\ \underbrace{1}_{B_3}$$

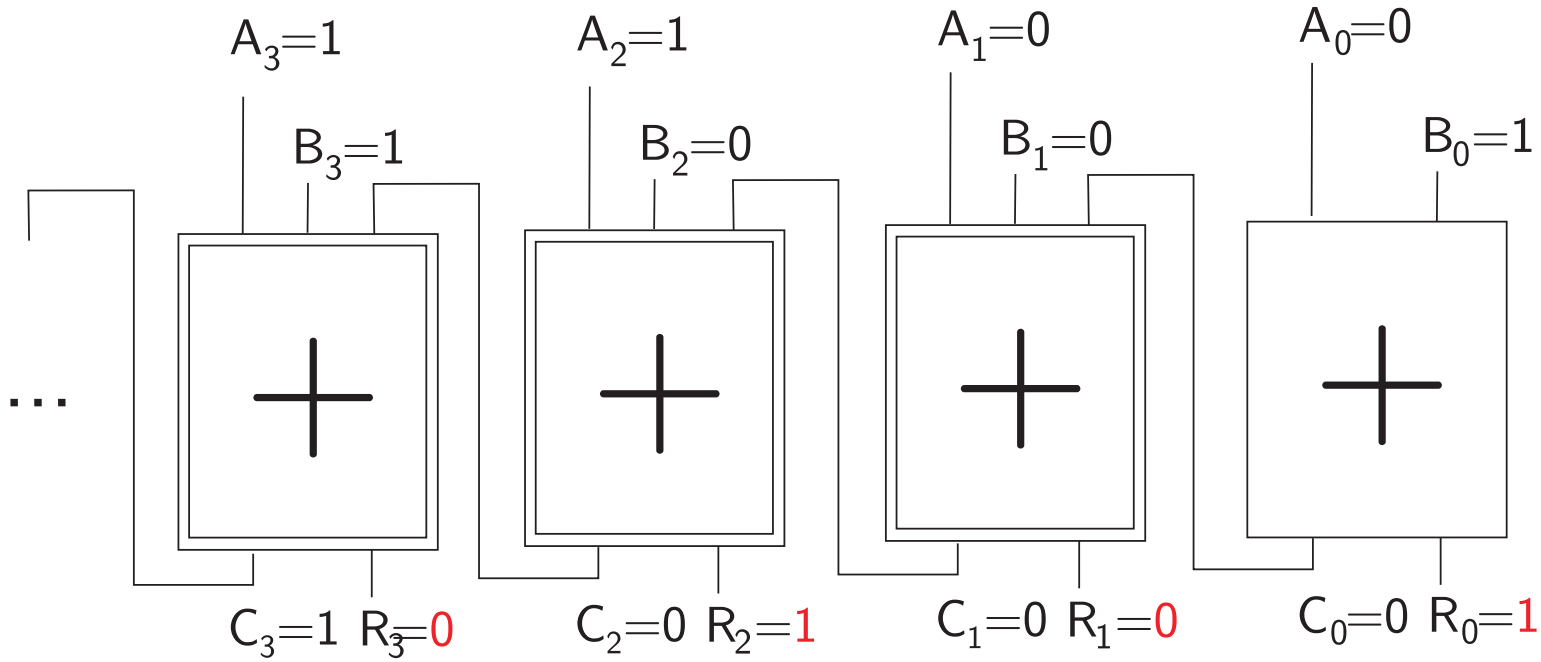
e inseriamoli come ingressi nei quattro sommatori:



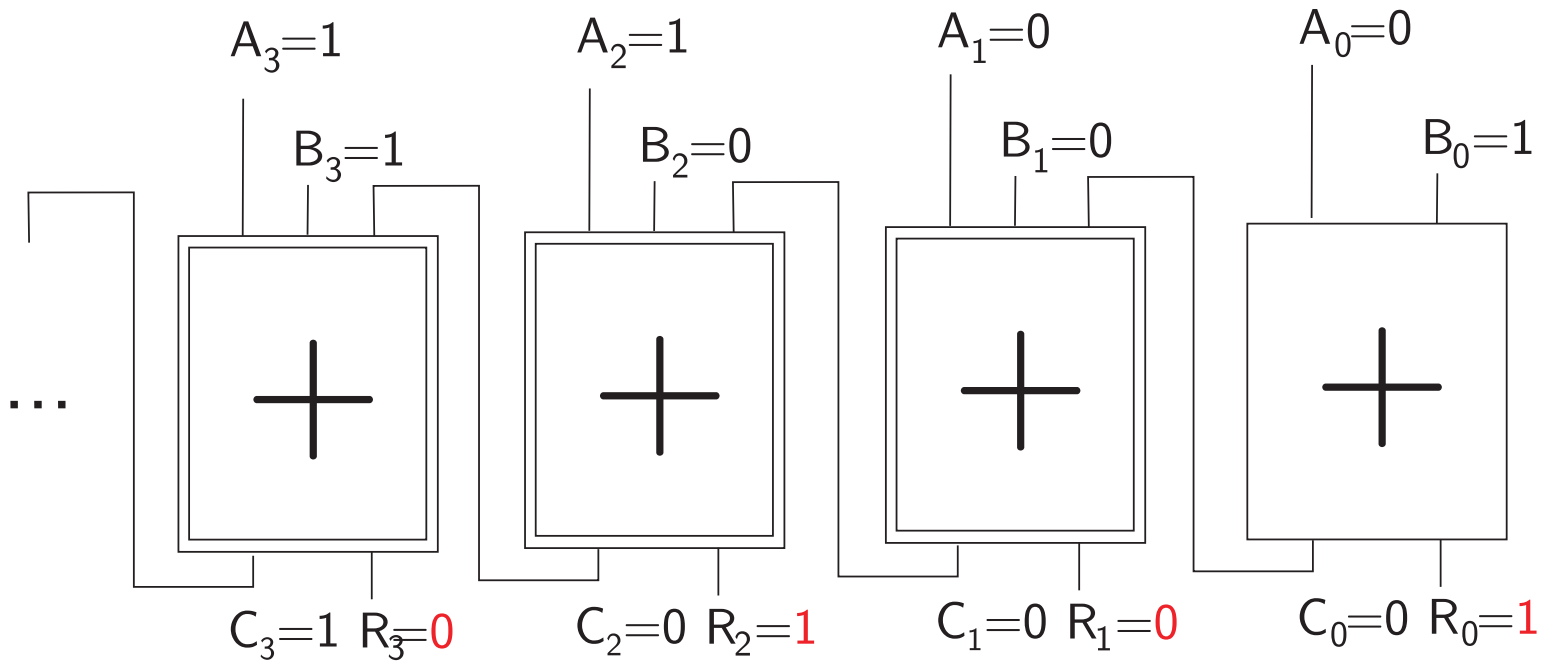
(al contrario perché al solito noi scriviamo i numeri come gli Arabi)

ed ecco il risultato:

ed ecco il risultato:



ed ecco il risultato:



Considerato che il riporto finale è come un 1 sul quinto bit (che è 16 in decimale), il risultato finale è 10101, che corrisponde a 21 in decimale, come deve essere.

Disponendo quindi in serie tante "macchinette" simili a queste si possono eseguire le addizioni. Per esempio, con 32 sommatori composti si arriva a sommare addendi fino a 2 147 483 648. In ogni caso a noi interessa capire il principio, dopodiché spetta alla Tecnica trovare la soluzione migliore (neanche il sommatore che abbiamo indicato in realtà si fa così, perché è sconveniente usare come "mattoni" $\otimes, \oplus$ e la negazione, però l'isomorfismo con la Logica resta valido). La miniaturizzazione dei circuiti rende poi velocissimi questi calcoli e pone le basi per la realizzazione dei moderni elaboratori elettronici.