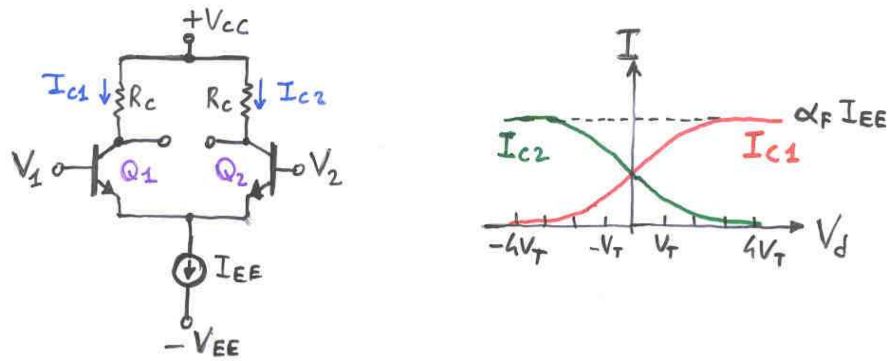
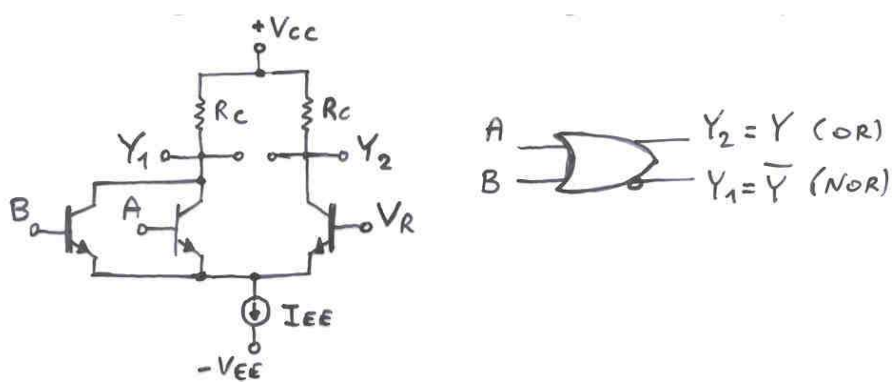
Circuiti ECL

- **La famiglia ECL utilizza commutazioni in regime non saturato e fornisce i circuiti con le più elevate velocità. L'elemento base è la coppia differenziale, sensibile alla differenza di tensione tra i due ingressi,** $V_d = V_1 - V_2$**:**



- **le due tensioni di uscita sono** $V_{o1} = V_{CC} - R_C I_{C1}$ **e** $V_{o2} = V_{CC} - R_C I_{C2}$**; se** $|V_d| \geq 4kT/q_e = 0.1$ **V si ha** $V_{o1} = HIGH$ **e** $V_{o2} = LOW$ **o viceversa, quindi le due tensioni di uscita sono complementari; si fissa poi** $V_2 = V_R$ **e si utilizza** V_1 **come ingresso**

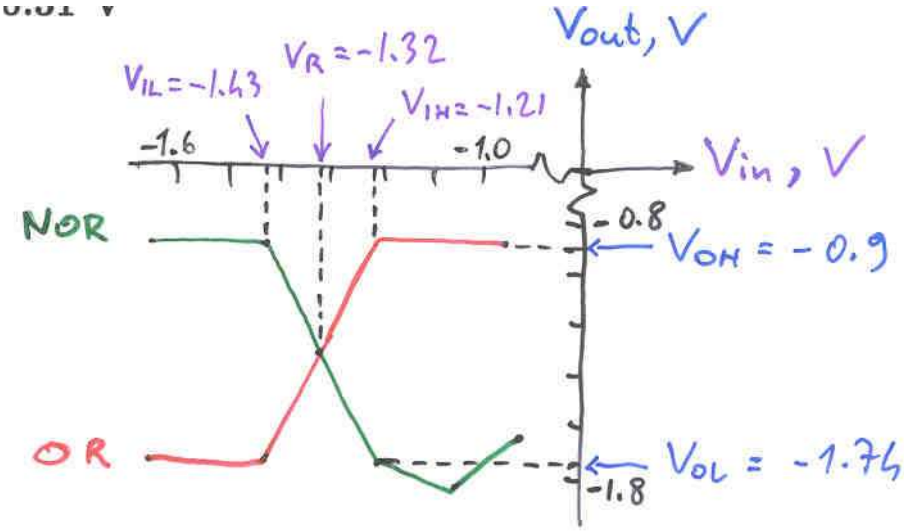
- **la porta logica fondamentale della famiglia ECL è l'OR/NOR:**



- **se uno o entrambi gli ingressi A e B sono a tensione** $V_{in} > V_R + 0.1$ **V si ha** $V_{o1} = LOW$ **e** $V_{o2} = HIGH$**; viceversa se A e B sono a tensione** $V_{in} < V_R - 0.1$ **V si ha** $V_{o1} = HIGH$ **e** $V_{o2} = LOW$**, quindi** Y_2 **è l'uscita OR mentre** Y_1 **è l'uscita NOR**

- **conviene poi ridurre il numero di alimentazioni necessarie (**$+V_{CC}$**,** $-V_{EE}$**,** V_R**) ponendo** $V_{CC} = 0$ **V e ricavando la tensione di riferimento** V_R **da** $-V_{EE} = -5.20$ **V**

- **per ottenere un funzionamento corretto è necessario spostare verso valori di tensione più negativi (mediante un inseguitore di emettitore sull'uscita) i due livelli logici: in definitiva si ottiene $V_{OH} = -0.9$ V, $V_{OL} = -1.74$ V con margini di rumore di circa 0.31 V**



- **il consumo tipico di una porta ECL, dovuto alle correnti assorbite dalla coppia differenziale, dalla sorgente di V_R e da alcuni diodi di stabilizzazione, va da 24 a 40 mW, risp. per le famiglie ECL 10K e 100K**

- **il *fan-out* è limitato dal carico capacitivo (se questo è troppo grande, le specifiche di velocità ECL non possono essere rispettate) e vale circa 10**

- **per non degradare il tempo di salita (circa 1 ns), la trasmissione dei segnali ECL deve essere particolarmente curata, preferibilmente usando linee di trasmissione adattate a 50 Ω (la scala di distanze è data dalla velocità della luce nel semiconduttore, circa 15 cm/ns: per distanze corrispondenti a tempi paragonabili con il tempo di salita o superiori NON si possono usare semplici collegamenti non adattati)**

Codifica dell'informazione

- **informazione** *non numerica*, **ad es. codice sorgente di un programma scritto in linguaggio di alto livello: vanno codificati i** *simboli* **dell'alfabeto usato (ad es.** $A \ldots Z\ 0 \ldots 9\ (\ )\ +\ -\ *\ /$ **etc.) con una successione di cifre binarie 0 e 1, e viene usato universalmente il codice ASCII a 7 bit, in cui ad esempio:**

$$
\begin{array}{ccl}
/ & \rightarrow & 0101111_2 = 47_{10} = 2F_H \\
0 & \rightarrow & 0110000_2 = 48_{10} = 30_H \\
A & \rightarrow & 1000001_2 = 65_{10} = 41_H \\
a & \rightarrow & 1100001_2 = 97_{10} = 61_H
\end{array}
$$

- **informazione** *numerica*, **ad es. costante nel codice oggetto di un programma che dovrà essere utilizzata per calcoli in fase di esecuzione: in questo caso non conviene usare il codice ASCII, conviene invece usare la rappresentazione binaria naturale** $b_n \times 2^n + b_{n-1} \times 2^{n-1} + \ldots + b_1 \times 2^1 + b_0 \times 2^0$**; per esprimere un numero decimale in forma binaria si usano successive divisioni per 2:**

$$
\begin{array}{rcrl}
53/2 & = & 26 & \textit{resto } 1 \times 2^0 \quad (LSB) \\
26/2 & = & 13 & \textit{resto } 0 \times 2^1 \\
13/2 & = & 6 & \textit{resto } 1 \times 2^2 \\
6/2 & = & 3 & \textit{resto } 0 \times 2^3 \\
3/2 & = & 1 & \textit{resto } 1 \times 2^4 \\
1/2 & = & 0 & \textit{resto } 1 \times 2^5 \quad (MSB)
\end{array}
$$

  $\Rightarrow 53_{10} = 110101_2$

- **altre rappresentazioni numeriche:**
  - **esadecimale ovvero in base 16: gruppi di 4 bit (a partire dal bit meno significativo, LSB) vengono rappresentati con i simboli** $0 \ldots 9\ A \ldots F$**, ad es.:**

$$707_{10} = 10\ 1100\ 0011_2 = 2C3_H \qquad (21)$$

  - **BCD (Binary Coded Decimal): ciascuna cifra decimale viene codificata in forma binaria (4 bit), ad es.:**

$$137_{10} = 0001 - 0011 - 0111 \qquad (22)$$

    in questo modo si usano più bit (12) rispetto alla rappresentazione binaria naturale ($137_{10} = 10001001_2$)

Numeri positivi e negativi

- **Per effettuare calcoli numerici è necessario rappresentare numeri interi positivi e negativi, in questo ambito esistono tre diverse rappresentazioni:**

  1. **segno+valore assoluto: il bit più significativo (MSB) viene riservato al segno ($0 \rightarrow +$, $1 \rightarrow -$), gli altri bit al valore assoluto; viene usata per visualizzare i numeri, ma non è adatta per i calcoli: la sottrazione richiede un'implementazione diversa rispetto all'addizione.**
  2. **binaria con *offset*: nell'esempio (con 4 bit a disposizione) i numeri $-8\ \ -7\ \ \ldots\ \ +6\ \ +7$ vengono aumentati di 8 per trasformarli nella successione $0\ 1\ \ \ldots\ \ 14\ 15$; è la rappresentazione preferita per i contatori binari e per la conversione A/D e D/A, ma non è la migliore per i calcoli.**
  3. **complemento a 2: i numeri positivi sono in rappr. binaria naturale, mentre i negativi si ottengono facendo il *complemento a 1* del corrisp. numero positivo e aggiungendo ancora 1, cioè ($b_n b_{n-1} \ldots b_2 b_1 \Rightarrow \overline{b_n}\ \overline{b_{n-1}} \ldots \overline{b_2}\ \overline{b_1} + 1$); in questo modo l'addizionatore binario può funzionare con numeri sia positivi (MSB=0) che negativi (MSB=1).**

DIVERSE RAPPRESENTAZIONI DI INTERI
CON 4 BIT A DISPOSIZIONE

numero	segno + valore ass.	binaria con offset	complemento a 2
+7	0111	1111	0111
+6	0110	1110	0110
+5	0101	1101	0101
+4	0100	1100	0100
+3	0011	1011	0011
+2	0010	1010	0010
+1	0001	1001	0001
0	0000	1000	0000
-1	1001	0111	1111
-2	1010	0110	1110
-3	1011	0101	1101
-4	1100	0100	1100
-5	1101	0011	1011
-6	1110	0010	1010
-7	1111	0001	1001
-8	-	0000	1000
(-0)	1000	-	-

- **verifichiamo che addizione e sottrazione funzionino correttamente nella rappresentazione complemento a 2:**

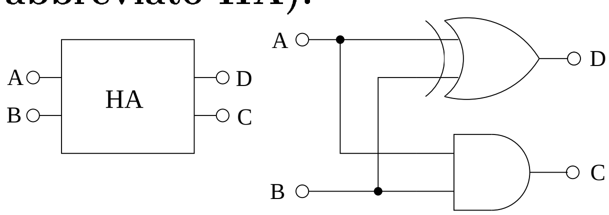
5+(-2):	**0101**	**(+5)**	**2-5:**	**0010**	**(+2)**	**5+(-5):**	**0101**	**(+5)**
	1110	**(-2)**		**1011**	**(-5)**		**1011**	**(-5)**
riporto	**0011**	**(+3)**		**1101**	**(-3)**	*riporto*	**0000**	**(0)**

- **questo significa che possiamo usare essenzialmente lo stesso circuito sommatore, che realizza l'addizione bit a bit con riporto, per effettuare somme e sottrazioni; per effettuare la sottrazione $A - B$ bisogna prima trasformare B nel suo complemento a 2 ($\overline{B} + 1$) con un circuito supplementare**

- **la presenza del *riporto* (bit successivo al MSB, ovvero di peso 2^4 negli esempi) indica che il risultato è positivo o nullo; bisogna però notare che $5+5 = -6$, dato che $+10$ non è rappresentabile con soli (3+1) bit**

Circuiti sommatori

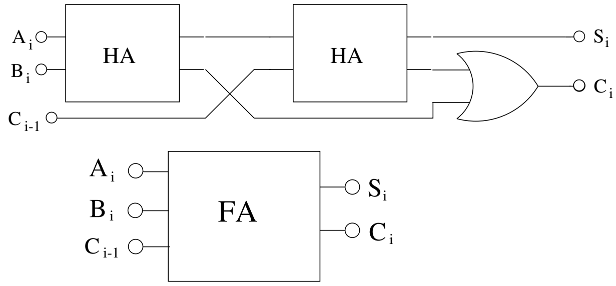
- **il primo circuito da considerare è il semisommatore (*half-adder*, abbreviato HA):**



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

Digit $D = \overline{A}B + A\overline{B}$
Carry $C = AB$

- **per effettuare l'operazione di somma a più bit è necessario tener conto dell'eventuale riporto dal bit di peso minore, il che ci porta al sommatore completo (*full-adder*, abbreviato FA) a 1 bit, che può essere costruito ad es. combinando due semisommatori :**



A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- **analizzando le espressioni booleane (nella cosiddetta *forma canonica*) delle uscite di somma (S_i) e di riporto (C_i) in un sommatore completo:**

$$S_i = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1} \quad (23)$$
$$C_i = \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} + A_iB_i\overline{C}_{i-1} + A_iB_iC_{i-1} \quad (24)$$

  si trova che il riporto può essere espresso in forma più semplice (esso è 1 se almeno due dei tre bit in ingresso sono 1):

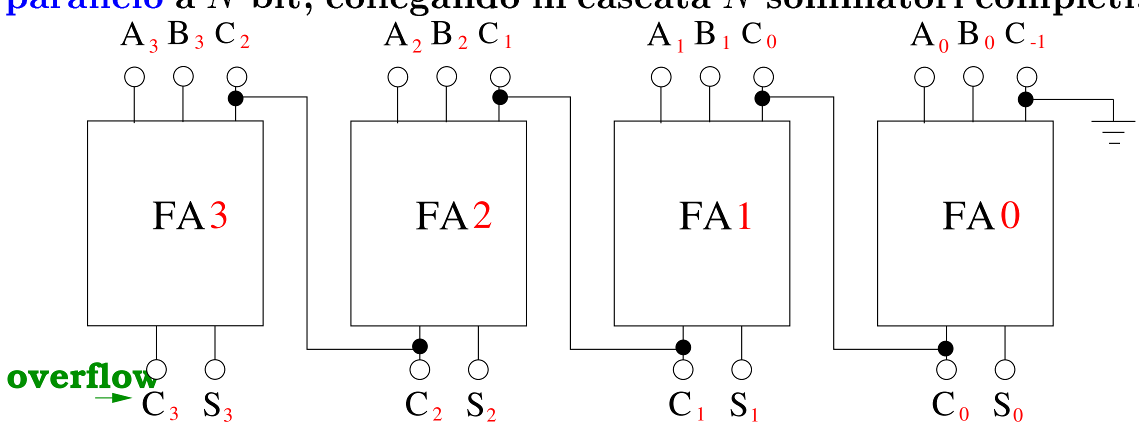
$$C_i = B_iC_{i-1} + C_{i-1}A_i + A_iB_i \quad (25)$$

  e analogamente per la somma si trova una forma più semplice (che utilizza anche la negazione di C_i, ovvero del riporto di ordine i):

$$S_i = A_i\overline{C}_i + B_i\overline{C}_i + C_{i-1}\overline{C}_i + A_iB_iC_{i-1} \quad (26)$$

  queste espressioni più semplici portano alla realizzazione del sommatore completo a 1 bit mediante porte AOI (And-Or-Invertitore), preferita nei circuiti integrati.

- **a questo punto è possibile costruire un sommatore binario parallelo a N bit, collegando in cascata N sommatori completi:**



- **va notato che la massima frequenza di funzionamento è limitata dal tempo di propagazione del *Carry* attraverso tutti i bit; inoltre è possibile collegare in cascata due sommatori paralleli a 4 bit per ottenerne uno a 8 bit e così via.**

- **Sono disponibili sommatori a 1, 2 e 4 bit nella categoria MSI (*Medium Scale Integration*): ad es. il sommatore a 4 bit 54LS283 (contenente circa 200 componenti) fornisce il risultato in circa 16 ns e consuma 190 mW.**