Circuiti Combinatori e Unità Aritmetico-Logiche

Decoder, Multiplexer, ALU e Circuiti Aritmetici

Prof. Fedeli Massimo

IIS Fermi Sacconi Ceci - Ascoli Piceno

5 gennaio 2026

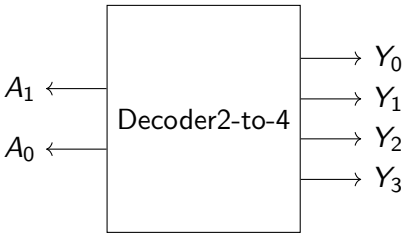
Indice degli Argomenti

1. Circuiti di Selezione e Distribuzione
2. Codificatori
3. Unità Logiche
4. Circuiti Aritmetici

Decodificatore (Decoder)

Definizione:

- Circuito con n ingressi e 2^n uscite
- Attiva una sola uscita alla volta
- L'uscita attivata corrisponde al valore binario dell'ingresso



Applicazioni:

- Decodifica indirizzi di memoria
- Selezione dispositivi
- Conversione binario-decimale

Tavola di verità 2-to-4:

A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

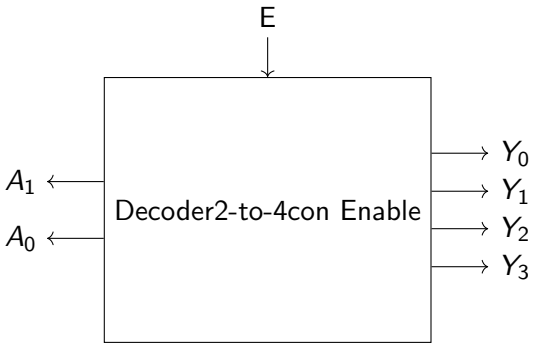
Decodificatore con Enable

Ingresso Enable (E):

- Abilita/disabilita il decodificatore
- Se E=0: tutte le uscite a 0
- Se E=1: funzionamento normale

Equazioni logiche:

$$Y_0 = E \cdot \overline{A_1} \cdot \overline{A_0}$$
$$Y_1 = E \cdot \overline{A_1} \cdot A_0$$
$$Y_2 = E \cdot A_1 \cdot \overline{A_0}$$
$$Y_3 = E \cdot A_1 \cdot A_0$$



Esempio applicativo:
Selezione banco di memoria:

- 4 banchi di memoria
- Enable seleziona quale banco attivare

Demultiplatore (DEMUX)

Definizione:

- Circuito con 1 ingresso dati
- n ingressi di selezione
- 2^n uscite
- Indirizza il dato verso una delle uscite

Funzionamento:
Il valore di D viene trasferito sull'uscita Y_i selezionata dagli ingressi di selezione, mentre tutte le altre uscite rimangono a 0.

Relazione con Decoder:
Un DEMUX è equivalente a un Decoder dove l'Enable diventa l'ingresso dati.

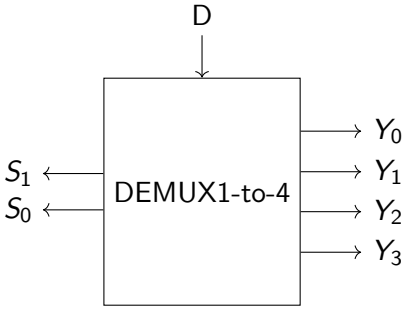


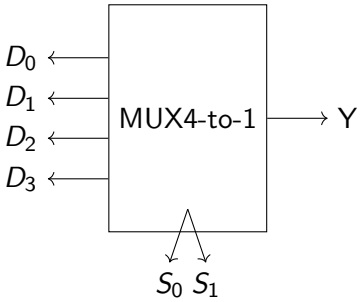
Tavola di verità (D=1):

S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Multiplatore (MUX)

Definizione:

- 2^n ingressi dati
- n ingressi di selezione
- 1 uscita
- Seleziona uno degli ingressi e lo trasferisce all'uscita



Applicazioni:

- Selezione sorgenti dati
- Implementazione funzioni logiche
- Routing segnali
- Multiplexing temporale

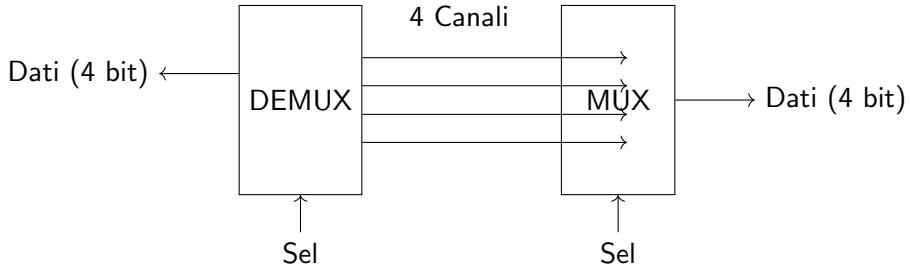
Tavola di verità:

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2

Equazione MUX 4-to-1:

Demultiplatori e Multiplatori in Parallelo

Trasmissione dati multi-bit:



Caratteristiche:

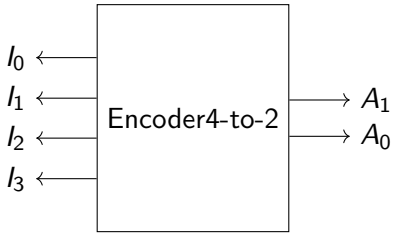
- Trasmissione simultanea di più bit
- Stessi segnali di selezione per tutti i MUX/DEMUX
- Utilizzato in bus dati e comunicazioni

Esempio: Bus dati a 8 bit

Codificatore Binario (Encoder)

Definizione:

- Funzione opposta al decoder
- 2^n ingressi
- n uscite
- Converte posizione attiva in codice binario



Limitazione:
Può essere attivo un solo ingresso alla volta.
Se più ingressi sono attivi, il risultato è imprevedibile.

Tavola di verità:

I_3	I_2	I_1	I_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Equazioni Encoder 4-to-2:

$$A_1 = I_2 + I_3$$

Applicazioni:

Codificatore di Priorità (Priority Encoder)

Problema risolto:
Gestisce più ingressi attivi simultaneamente assegnando priorità.

Funzionamento:

- Priorità all'ingresso con indice più alto
- Se $I_3 = 1$, uscita $= 11$ (indipendentemente dagli altri)
- Se $I_3 = 0$ e $I_2 = 1$, uscita $= 10$
- E così via...

Uscite aggiuntive:

- **GS** (Group Select): indica se almeno un ingresso è attivo
- **EO** (Enable Output): valida l'uscita

Tavola di verità 4-to-2:

I_3	I_2	I_1	I_0	A_1	A_0	GS
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Applicazioni critiche:

- Gestione interrupt nei microprocessori
- Arbitraggio bus
- Sistemi con richieste multiple

Esempio: Sistema con 4 periferiche
CPU priorità più alta → disco → tastiera → mouse

Unità Logiche (Logic Unit)

Definizione:
Circuito che esegue operazioni logiche bit a bit su due operandi.

Operazioni principali:

- AND ($A \wedge B$)
- OR ($A \vee B$)
- XOR ($A \oplus B$)
- NOT ($\overline{A}$)
- NAND, NOR, XNOR

Struttura:

- Ingressi: due operandi (A, B)

Implementazione con MUX:

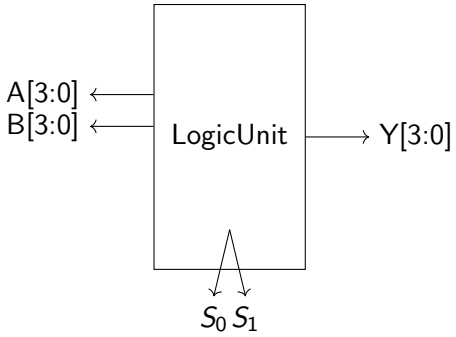


Tavola operazioni (per bit):

S_1	S_0	Operazione
0	0	$Y = A \wedge B$
0	1	$Y = A \vee B$

Circuiti di Scorrimento (Shifter)

Tipi di scorrimento:

1. **Scorrimento Logico:**
   - **Sinistra (SHL):** inserisce 0 a destra
   - **Destra (SHR):** inserisce 0 a sinistra
   - Esempio: 1011 SHL $\rightarrow$ 0110
   - Esempio: 1011 SHR $\rightarrow$ 0101
2. **Scorrimento Aritmetico:**
   - **Sinistra (SAL):** = SHL
   - **Destra (SAR):** mantiene il bit di segno
   - Esempio: 1011 SAR $\rightarrow$ 1101
   - Usato per numeri con segno
3. **Rotazione (Rotate):**
   - **ROL:** bit uscente entra a destra
   - **ROR:** bit uscente entra a sinistra
   - Esempio: 1011 ROL $\rightarrow$ 0111
   - Nessun bit perso
4. **Rotazione con Carry:**
   - **RCL:** coinvolge flag carry
   - **RCR:** per operazioni estese

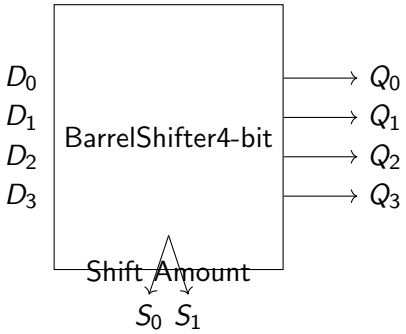
Applicazioni matematiche:

- SHL di 1 posizione = moltiplicazione per 2

Implementazione Barrel Shifter

Barrel Shifter:

Circuito che esegue scorrimenti di n posizioni in un solo ciclo.



Esempio di funzionamento (4-bit):

S_1	S_0	Q_3	Q_2	Q_1	Q_0	Shift
0	0	D_3	D_2	D_1	D_0	0 posizioni
0	1	D_2	D_1	D_0	0	1 posizione

Semisommatore (Half Adder)

Definizione:
Circuito che somma 2 bit producendo somma e riporto.

Ingressi:

- A, B (bit da sommare)

Uscite:

- S (Sum - somma)
- C (Carry - riporto)

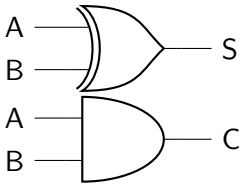
Equazioni:

$$S = A \oplus B$$
$$C = A \cdot B$$

Tavola di verità:

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

Schema logico:



Limitazione:

Sommatore Completo (Full Adder)

Definizione:
Somma 3 bit (2 operandi + riporto precedente)

Ingressi:

- A, B (bit da sommare)
- C_{in} (riporto ingresso)

Uscite:

- S (somma)
- C_{out} (riporto uscita)

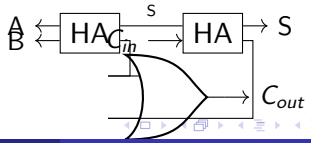
Equazioni:

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + C_{in}(A \oplus B)$$

Tavola di verità:

A	B	C_{in}	S	C_{out}
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

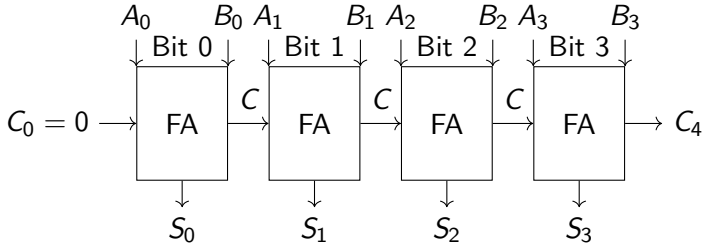
Implementazione con 2 Half Adder:



Addizionatore a Propagazione (Ripple Carry Adder)

Principio:

Collegamento in cascata di Full Adder per sommare numeri multi-bit.



Caratteristiche:

- Semplice da implementare
- Il riporto si propaga da destra a sinistra
- **Tempo di propagazione:** $T = n \times t_{FA}$ dove n è il numero di bit

Svantaggio principale:

Sottrazione Binaria

Metodo diretto:
Prestito invece di riporto

Half Subtractor:

- Ingressi: A, B
- Uscite: D (differenza), B (prestito)

Equazioni:

$$D = A \oplus B$$
$$B_{out} = \overline{A} \cdot B$$

Full Subtractor:

Metodo preferito:
Usare complemento a due

$$A - B = A + (\text{complemento di } B) + 1$$

Vantaggi:

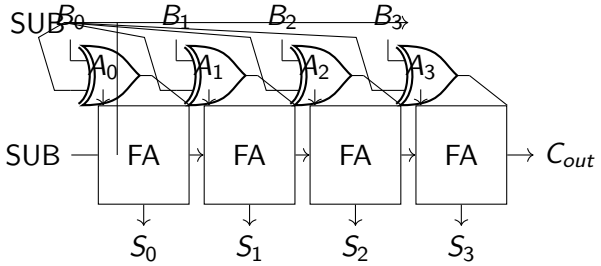
- Riutilizza circuito addizionatore
- Più efficiente
- Un solo tipo di circuito per + e -

Esempio: $5 - 3$ (4 bit)

$$5 = 0101$$
$$3 = 0011$$

Circuito Somma e Sottrazione

Implementazione unificata:



Funzionamento:

- **SUB = 0:** Addizione ($S = A + B$)
  - XOR lascia passare B invariato
  - $C_{in} = 0$
- **SUB = 1:** Sottrazione ($S = A - B$)
  - XOR inverte tutti i bit di B ($\overline{B}$)

Addizionatore a Riporto Anticipato (Carry Lookahead)

Problema del Ripple Carry:
Ritardo proporzionale al numero di bit.

Soluzione: Calcolare tutti i riporti in parallelo

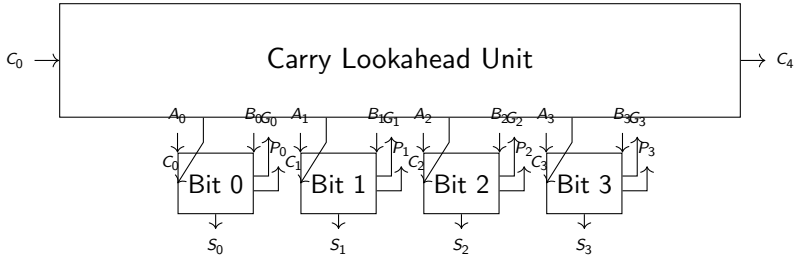
Definizioni:

- **Generate (G):** $G_i = A_i \cdot B_i$ (genera sempre riporto)
- **Propagate (P):** $P_i = A_i \oplus B_i$ (propaga riporto)

Equazioni riporto (4 bit):

$$C_0 = \text{Carry-in}$$
$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$$
$$C_3 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$$
$$C_4 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0$$

Struttura del Carry Lookahead Adder



Confronto prestazioni (32 bit):

Tipo	Ritardo	Complessità
Ripple Carry	$32 \times t_{FA}$	Bassa
Carry Lookahead	$\sim 4 \times t_{FA}$	Alta

Approccio gerarchico:

Rappresentazione in Complemento a Due

Definizione:
Metodo standard per rappresentare numeri con segno.

Procedura:

1. Invertire tutti i bit (complemento a uno)
2. Aggiungere 1 al risultato

Esempio: -5 in 8 bit

$$+5 = 00000101$$
$$\text{Inverti} = 11111010$$
$$+1 = 11111011 = -5$$

Riconoscimento segno:

Range (n bit):

$$-2^{n-1} \text{ a } +2^{n-1} - 1$$

Esempio 4 bit: da -8 a +7

Binario	Decimale
0111	+7
0110	+6
...	...
0001	+1
0000	0
1111	-1
1110	-2
...	...
1000	-8

Vantaggi:

Rilevamento Overflow

Overflow:
Si verifica quando il risultato non può essere rappresentato con il numero di bit disponibili.

Condizioni overflow (complemento a due):

- Somma di due positivi dà negativo
- Somma di due negativi dà positivo

Formula rilevamento:

$$V = C_n \oplus C_{n-1}$$

dove C_n è il riporto finale e C_{n-1} il riporto del bit di segno.

Esempi (4 bit):

Overflow presente:

$$\begin{array}{rll} & 0110 & (+6) \\ + & 0101 & (+5) \\ \hline & 1011 & (-5) \quad \text{ERRORE!} \end{array}$$

Overflow assente:

$$\begin{array}{rll} & 0011 & (+3) \\ + & 0010 & (+2) \\ \hline & 0101 & (+5) \quad \text{OK} \end{array}$$

Moltiplicazione Binaria - Metodo Carta e Penna

Principio:
Simile alla moltiplicazione decimale

Esempio: $6 \times 5 = 30$ (4 bit)

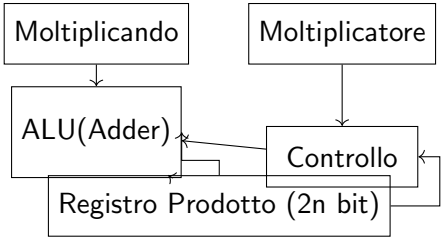
				0	1	1		0	(6)
$\times$				0	1	0		1	(5)
				0	1	1	0		(6×1)
			0	0	0	0			(6×0)
		0	1	1	0				(6×1)
$+$	0	0	0	0					(6×0)
	0	0	1	1	1	1		0	(30)

Osservazioni:

- Prodotto di numeri a n bit richiede $2n$ bit
- Se moltiplicatore bit $= 1 \rightarrow$ somma moltiplicando
- Se moltiplicatore bit $= 0 \rightarrow$ somma 0

Moltiplicatore Hardware

Implementazione sequenziale:



Fasi dell'algoritmo (per n bit):

1. Esamina LSB del moltiplicatore
2. Se LSB $= 1$: Prodotto $=$ Prodotto $+$ Moltiplicando
3. Shift destro del moltiplicatore
4. Shift sinistro del moltiplicando
5. Ripeti per n volte

Divisione Binaria

Algoritmo di divisione (simile a divisione decimale):

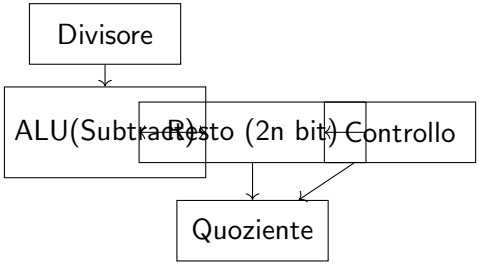
Esempio: $13 \div 3$ in binario

	0100	(quoziente = 4)
0011	1101	(dividendo = 13)
	0011	(sottrai divisore)
	1010	
	0011	
	0111	
	0011	
	0100	
	0011	
	0001	(resto = 1)

Algoritmo sequenziale:

1. Inizializza resto = dividendo

Divisore Hardware



Caratteristiche:

- Operazione più lenta delle altre aritmetiche
- Richiede n iterazioni per n bit
- Gestione divisione per zero essenziale

Divisione con segno:

- Convertire operandi in valori assoluti
- Eseguire divisione senza segno

Comparatore di Magnitudine

Funzione:
Confronta due numeri binari A e B

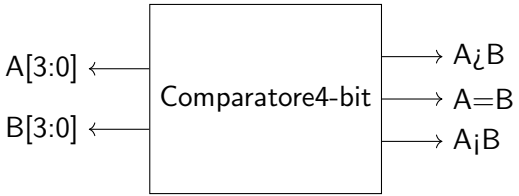
Uscite:

- A ¿ B
- A = B
- A ¡ B

Comparatore 1-bit:

$$A = B : A \odot B = \overline{A \oplus B}$$
$$A > B : A \cdot \overline{B}$$
$$A < B : \overline{A} \cdot B$$

Metodo alternativo:



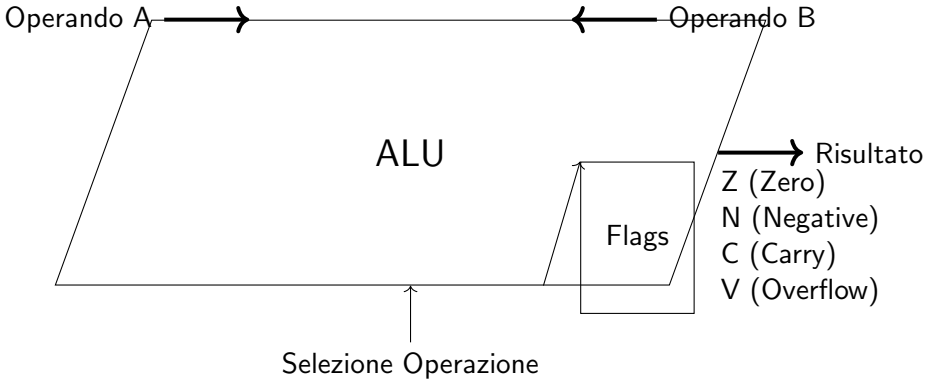
Comparatore n-bit a cascata:
Confronto bit per bit da MSB a LSB

1. Confronta bit più significativi
2. Se uguali, passa al bit successivo
3. Se diversi, determina risultato

Equazioni 4-bit:

Unità Aritmetico-Logica (ALU)

Componente centrale del processore:



Operazioni tipiche:

Aritmetiche:

- ADD, SUB

Logiche:

- AND, OR, XOR, NOT

Esempio: ALU 74181

ALU storica a 4 bit (1970):

Caratteristiche:

- 4 bit di dati
- 5 bit di selezione (S_0 - S_4)
- Modo aritmetico/logico (M)
- 16 funzioni logiche
- 16 funzioni aritmetiche

Ingressi:

- A[3:0], B[3:0]: operandi
- S[3:0]: selezione funzione
- M: modo (0=aritm, 1=logic)
- C_n: carry-in

Uscite:

- F[3:0]: risultato
- A=B: uguaglianza
- C_{n+4}: carry-out
- G, P: generate/propagate

Alcune operazioni (M=0):

S[3:0]	Operazione
0000	A
0110	A - B
1001	A + B
1100	A + 1

Espandibilità:

Applicazioni dei Circuiti Combinatori

1. **Processori:**
   - ALU per tutte le operazioni aritmetico-logiche
   - Decoder per decodifica istruzioni
   - MUX per selezione operandi e risultati

2. **Memorie:**
   - Decoder per selezione indirizzo
   - MUX per accesso dati
   - Comparatori per gestione cache

3. **Comunicazioni:**
   - MUX/DEMUX per multiplexing segnali
   - Encoder per codifica dati
   - Comparatori per rilevazione errori

4. **Sistemi embedded:**
   - Priority encoder per gestione interrupt

Riepilogo e Conclusioni

Concetti chiave appresi:

1. **Circuiti di routing:**
   - Decoder, Encoder, MUX, DEMUX
   - Selezione e distribuzione segnali
2. **Circuiti logici:**
   - Operazioni bit a bit
   - Scorrimenti e rotazioni
3. **Circuiti aritmetici:**
   - Addizionatori (Ripple Carry, Carry Lookahead)
   - Moltiplicatori e divisori
   - Complemento a due e gestione overflow
4. **ALU:**
   - Combinazione di tutti i circuiti
   - Cuore computazionale del processore

Prospettive future: