Rappresentazione elettronica dell'informazione

Necessita' di rappresentare l'informazione

Le macchine hanno lo scopo di agevolare la vita dell'uomo mediate operazioni automatizzate. L'automatica e' una scienza che si occupa di studiare e realizzare macchine in grado di trasformare materie prime in prodotti finiti senza l'intervento diretto dell'uomo. L'informatica e' una branca dell'automatica dove le materie prime e i prodotti finiti sono dati e informazioni.

Una macchina automatica e' un oggetto in grado di modificare in modo autonomo il proprio stato agendo su una o piu' grandezze fisiche (posizione, peso, temperatura). Una macchina informatica necessita di una grandezza fisica che possa essere modificata in modo automatico e rappresentare concetti astratti.

Per rappresentare le informazioni dobbiamo:

- Identificare l'informazione che vogliamo elaborare: **il rappresentato**.
- Scegliere una grandezza fisica per il rappresentato: **il rappresentate**.
- Definire una **corrispondenza** tra rappresentato e rappresentante.

Il rappresentato puo' essere qualsiasi informazione che richieda un'elaborazione automatica e da' un'informazione concettuale di un fenomeno fisico (velocita' di un'automobile, temperatura di un locale, ammontare di euro in un conto bancario).

Il rappresentante e' una grandezza fisica il cui comportamento e' noto e modificabile (di tipo meccanico per il tachimetro dell'automobile, termico per valutare la temperatura).

La scelta del rappresentante e' dunque una componente importante per poter accedere al dato e i rappresentanti elettrici sono tra i migliori canditati per dimensioni, velocita' e costi di realizzazione (in particolare quelli basati su silicio).

Rappresentazione analogica dell'informazione

Esempio di corrispondenza tra rappresentato e rappresentante: Come rappresentato prendiamo il valore numerico della temperatura di un locale: che sara' variabile tra due valori Tmin e Tmax. Come rappresentate scegliamo la tensione elettrica tra due punti di un circuito: che sara' variabile tra due valori 0 e Vmax. La corrispondenza piu' ovvia e' **biunivoca** e lineare (si dice analogica perche' abbiamo una completa analogia anche su piccole variazioni).

La rappresentazione analogica e':

- **Fedele**: in quanto piccole variazioni del rappresentato comportano piccole variazioni sul rappresentante.
- **Intuitiva**: poiche' dal rappresentate si risale immediatamente al rappresentato e le variazioni del rappresentate fanno capire immediatamente il

comportamento del rappresentato.
- **Vulnerabile**: perche' ogni valore del rappresentante e' ammissibile dunque ogni variazione alla curva di corrispondenza dovuta a disturbi o invecchiamenti, genera errori di rappresentazione. Poiche' ogni rappresentazione e' intrinsecamente affetta da errori di approssimazione, qualora il rappresentante perda la corretta taratura la propagazione e la moltiplicazione dell'errore, quel rappresentante diventa poco affidabile o addirittura non significativo.

Rappresentazione digitale dell'informazione

Nasce per eliminare la vulnerabilita' della rappresentazione analogica, evitando che qualsiasi valore sia ammissibile, tramite la scelta di un numero discreto di valori del rappresentante.

La corrispondenza tra rappresentato e rappresentante e' **univoca**:

- **Robusta**: una variazione del valore del rappresentate dovute a disturbi o invecchiamento vengono riconosciuti come **errori**.
- **Munita di autocorrezione**: se le variazioni del rappresentato sono sufficientemente piccole si puo' risalire ad un valore corretto del rappresentante.
- **Meno intuitiva**: ad un valore del rappresentato corrisponde un valore del rappresentato, ma non viceversa.
- **Poco fedele**: le variazioni di un rappresentato all'interno di un intervallo, non vengono rappresentate.

Rappresentazione binaria

- **Estremamente robusta**: due sole cifre agli estremi del campo di variabilita' del rappresentate.
- **Autocorrezione molto semplice**: tramite arrotondamento.
- **Economica**: circuiti binari consumano molto poco, dunque e' possibile impiegarne molti in una singola applicazione.
- **Troppo limitata**: non e' possibile fare fede unicamente a due valori, serve un aumento di accuratezza. Impieghiamo dunque **piu' rappresentati** ad un rappresentato di modo che possano fornirci stringhe di bit (invece di un bit solo) e migliorare l'accuratezza in maniera esponenziale.

Rappresentazione di informazioni con varie quantita' di bit

Codifiche binarie di valori numerici

Tramite una rappresentazione di tipo binario possiamo rappresentare numeri, caratteri, matrici, forme d'onda. Ci rifacciamo alla notazione decimale di tipo

posizionale, dove ogni cifra ha un determinato **peso** dato dalla sua posizione nel numero. Nella notazione decimale il peso e' una potenza di 10 (base della notazione). Anche le altre rappresentazioni, come quella binaria e quella esadecimale sono di tipo posizionale.

Per rappresentare una determinata informazione servono un numero di cifre:

> 0 <= n <= (baseˆc)-1 Dove **base** sta ad indicare il numero di cifre possibili per la determinata notazione (es. 10 per la notazione decimale, 2 per quella binaria, ecc) e **c** e' il numero di cifre che voglio impiegare per la rappresentazione.

Dunque per rappresentare un numero n ho bisogno di almeno:

> c = log(base10)n

Per quanto riguarda i **numeri frazionari** possiamo ricorrere anche alle potenze negative e porre una virgola anche nella rappresentazione binaria. Per avere una buona precisione o estensione pero' serve una grande quantita' di bit.

Anche i numeri in **virgola mobile** vengono rappresentati mediante notazione esponenziale:

> n = m x bˆe **m** e' la mantissa costituita da cifre intere. **b** e' la base (10, 2, 16 a seconda del tipo di notazione). **e** e' l'esponente (positivo o negativo) da dare alla base per definire il peso delle cifre della mantissa. Serve dunque a muovere la posizone della virgola rispetto alle cifre della mantissa e consente di rappresentare valori molto piccoli o molto grandi.

Con i **numeri relativi** associamo un bit al segno (0 per i positivi, 1 per i negativi) e il resto dei bit al **modulo** del numero.

Codifica di numeri in complemento a 2

La notazione in modulo e segno ha due difetti:

- Ogni operazione di somma e sottrazione dipende dal segno degli operandi:
  - Se sono **concordi** si procede;
  - Se sono **discordi** si devono scambiare le operazioni;
  - Nel caso della sottrazione si deve fare un **confronto** per stabilire minuendo e sottraendo;
- Esistono due rappresentazioni per lo zero:
  - Zero+;
  - Zero-;

La rappresentazione in complemento a due e' una modalita' di rappresentazione per i numeri interi o a virgola fissa. Consiste nel rappresentare i numeri utiliz-

zando **il bit piu' significativo per indicarne il segno**, mentre gli altri per il modulo.

Per un totale di n bit e' possibile rappresentare l'intervallo: -2^(n-1) <= x <= 2^(n-1)-1. Si tratta di un intervallo di rappresentazione **asimmetrico**.

Per cambiare di segno un numero, basta **invertire tutti i suoi bit** e **sommare 1**. (A parte una singola eccezione: quando, in 8 bit, si converte il numero -128 si ottiene lo stesso numero in rappresentazione binaria (1000 0000). Questo perche' il maggior numero rappresentabile con 7 bit e' 127. Viene segnalato un overflow ma la sua rappresentazione e' valida.)

Le operazioni aritmetiche vanno effettuate rappresentando i numeri con lo stesso numero di bit. Il risultato verra' indicato anche lui con il numero dei bit dei suoi operandi e con il segno corretto. Quando si eseguono delle operazioni aritmetiche e' necessario prestare attenzione alla riga del **riporto** in quanto i due bit piu' significativi della suddetta riga contengono informazioni relative alla validita' dell'operazione: se sono identici (entrambi 1 o 0) allora l'operazione e' andata a buon fine, altrimenti si tratta di **overflow aritmetico**.

Questa notazione ha diversi **vantaggi**:

- La somma tra due numeri in complemento a due (a patto che il risultato cada nell'intervallo rappresentabile) avra' segno giusto di principio sia in caso di segni concordi che discordi.
- La sottrazione puo' avvenire come semplice somma con l'opposto del secondo operando.
- La semplicita' nel cambio di segno.
- Lo zero non ha due possibili rappresentazioni come nella notazione modulo e segno.

Ha pero' anche alcuni **svantaggi**:

- Per i numeri positivi si perde la **notazione posizionale** pesata che invece era presente nella notazione modulo e segno.
- L'intervallo di rappresentazione e' asimmetrico.
- Se l'operazione produce un risultato che non ricade nell'intervallo di rappresentazione per i bit disponibili si ha una situazione di **overflow arimetico**.

Codifiche binarie di informazioni non numeriche

Lettere e simboli sono rappresentabili con condifica binaria facendo riferimento al codice ASCII.

Le **immagini** sono una matrice rettangolare di pixel (punti immagine):

- Risoluzione spaziale: numero di pixel nella matrice.
- Risoluzione cromatica: numero di bit per pixel.

Il **segnale audio** e' una forma d'onda variabile nel tempo. Si campiona la forma d'onda e si memorizzano i campioni. La frequenza di campionamento deve essere almeno il doppio della massima frequenza del segnale (teorema di Nyquist-Shannon).

I **fimati** si ottiengono codificando separatamente il segnale audio e quello video (sequenza di immagini). Si utilizzano metodi di compressione.

Postulati dell'algebra booleana

La variabile booleana tratta solamente grandezze binarie. Le operazioni sono:

- **Negazione**, rappresentato con il meno (-)
- **Prodotto logico (and)**, rappresentato con un punto (.)
- **Somma logica (or)**, rappresentato con il simbolo piu' (+)

I 5 postulati dell'algebra booleana sono:

- P1: Questo postulato afferma solo che stiamo trattando grandezze binarie.
  - Definita X come grandezza variabile dell'algebra boolana avremo:
    - (P1) X=0 if X!=1;
    - (P1') X=1 if X!=0;
- P2: Questo postulato introduce un'operazione unaria (che si applica ad un solo operando con il cambio di segno) su grandezze booleane.
  - Definito -X il valore opposto assunto dalla variabile X (dove X e -X sono i due **letterali** della stessa variabile):
    - (P2) if X=0 then -X=1;
    - (P2') if X=1 then -X=0;
    - Si e' dunque definita l'operazione di **negazione (NOT)** dove X è il **negato** di -X.
- P3-P4-P5: definiscono operazioni binarie (che si applicano ad entrambi gli operandi).
  - (P3) 0.0=0;
  - (P3') 1+1=1;
  - (P4) 1.1=1;
  - (P4') 0+0=0;
  - (P5) 1.0=0.1=0;
  - (P5') 0+1=1+0=1;

Ogni postulato Pi dell'algebra booleana ha una seconda versione Pi' che si ottiene sostituendo:

- ogni valore 0 con un valore 1 (e viceversa)
- ogni operatore or con un operatore and (e viceversa) **Tutti i teoremi discendono dai postulati booleani devono godere di questa dualita'.**

I **Teoremi di De Morgan** stabiliscono le relazioni di equivalenza tra gli operatori logici and e or usando le negazioni:

-(X1 . X2 . X3) = -X1 + -X2 + -X3 -(X1 + X2 + X3) = -X1 . -X2 . -X3

Risulta particolarmente utile quando, durante la realizzazione su circuito degli operatori dell'algebra booleana, la tecnologia elettronica scelta favorisca la realizzazione dei sommatori logici, piuttosto che dei moltiplicatori logici o viceversa.

Esempio: **raccoglimento ad addendo comune** Se un operatore booleano compare piu' volte in un'espressione e' possibile raccoglierlo e quindi ottenere un'espressione equivalente.

> Sviluppiamo: 0: (A + B) . (A + C) 1: A.A + A.C + A.B + B.C 2: A . (1 + B + C) + (B.C) 3: A . (1) + (B.C) 4: A + (B.C)

La somma logica alla riga 2 (1 + B + C) avra' sempre come valore 1. Il prodotto logico alla riga 3 (A . 1) sara' sempre dipendente da A.

Aritmetica con l'algebra booleana

L'algebra per operare su grandezze bianrie opera su tre operatori: NOT, AND e OR. Consideriamo A e B come due numeri interi a 1 bit, quindi ciascuno dei numeri puo' valere 0 o 1.

A	B	Somma	Riporto
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Chiamiamo il nostro circuito **HALF ADDER** in quanto la somma e' parziale perche' non effettuiamo una rappresentazione del bit di riporto.

Riporto = A . B Somma = (A . -B) + (-A + B)

Se volessimo sommare due numeri A e B entrambi composti da 2 bit, avremo che entrambi i numeri possono avere i valori: 0, 1, 2, 3. Per fare cio' ci serviamo del circuito HALF ADDER e sommiamo le cifre meno significative A0 e B0 (dove abbiamo 2^0) e teniamo di conto il riporto (Riporto0)ottenuto dalla somma. Dopodiche' per effettuare il resto della somma, facciamo riferimento alla tabella di verita':

A1	B1	Riporto0	Somma1	Riporto1
0	0	0	0	0
0	0	1	1	0

A1	B1	Riporto0	Somma1	Riporto1
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

Riporto il metodo aritmetico per il calcolo rei valori di Riporto1 e Somma1 dove le coppie e le terne di valori fanno riferimento all'HALF ADDER. Riporto1 = A1B1 + A1R0 + B1R0 Somma1 = -A1-B1R0 + -A1B1-R0 + A1-B1-R0 + A1B1R0

Porte logiche

L'elettronica consente di realizzare nei circuiti integrati gli operatori fondamentali dell'algebra booleana (NOT, AND, OR, NAND, NOR).

Bistabili

Le porte logiche connesse con retroazioni danno luogo ad un comportamento che tiene memoria della storia nel circuito mediante i bistabili.

Sono circuiti in cui si raggiunge uno stato stabile con due possibili valori di uscita. I circuiti sono realizzati mediante operatori fondamentali dell'algebra booleana. Nei bistabili, quando attivati, ogni variazione di ingresso si ripescuote sulle uscite propagando al resto del circuito le conseguenze di queste variazioni (**trasparenza dei bistabili** noti come **LATCH**) dunque non filtrate.

E' possibile creare configurazioni **non trasparenti** mediante architetture come la **Master-Slave** dove le uscite non risentono immediatamente della variazione agli ingressi ma solo nel semiperiodo successivo al segnale di Clock aumentando il disaccoppiamento. I bistabili Master-Slave si chiamano **Flip-Flop**. Per realizzare un Flip-Flop bisogna connettere due LATCH a cascata utilizzando cosi' il segnale di controllo in **controfase**:

- C = 1 : Si attiva il LATCH Master, lo Slave rimane congelato;
- C = 0 : Si spegne il LATCH Master, si attiva lo Slave e l'output propaga.