La struttura dei sistemi embedded: periferiche, sensori e attuatori

Braione Pietro, revisione Domenico G. Sorrenti

Sistemi Embedded

Obiettivi

- Veloce revisione delle tecniche di gestione dei dispositivi periferici normalmente presenti nei sistemi embedded
- Timer, convertitori AD e DA, teorema del campionamento
- Sensoristica
- Attuazione (elettrica)

Periferiche

- Dispositivi che eseguono funzioni standard single-purpose
- Sensori e attuatori interfacciano il sottosistema computazionale con il sistema fisico
- In alcuni casi le funzionalità di una periferica potrebbero essere implementate programmando un processore general-purpose
- Occorre valutare i trade-off su complessità, dimensioni, consumi e costi del sistema

Comunicazioni con le periferiche

- Ne parliamo ora perché l'interazione con le periferiche avviene in maniera simile (o uguale, se ricordate) all'accesso in memoria
- Le periferiche si controllano scrivendo nei loro registri di controllo, e inviando/ricevendo dati nei/dai loro registri dati
- Come si accede a tali registri?
  - Periferiche mappate in memoria: una parte degli indirizzi dello spazio di indirizzamento viene non fatto decodificare alla memoria, ma è utilizzato (viene fatto decodificare) dalla periferica, per leggere/scrivere i suoi registri
  - Spazio di indirizzamento per l'I/O: con istruzioni assembly dedicate per accedervi (es. 8051 e x86)

Controllo di programma (polling)

- Il tipo di gestione di periferica più semplice, che richiede la minore complessità hardware e garantisce buone prestazioni a fronte di una totale dedizione della MCU al trasferimento.
- Utile nel caso di sistemi vincolati dai costi e che non debbano svolgere calcoli intensi e quei pochi calcoli dipendono da pochi dati ricevuti dalle periferiche.

Interruzione di programma

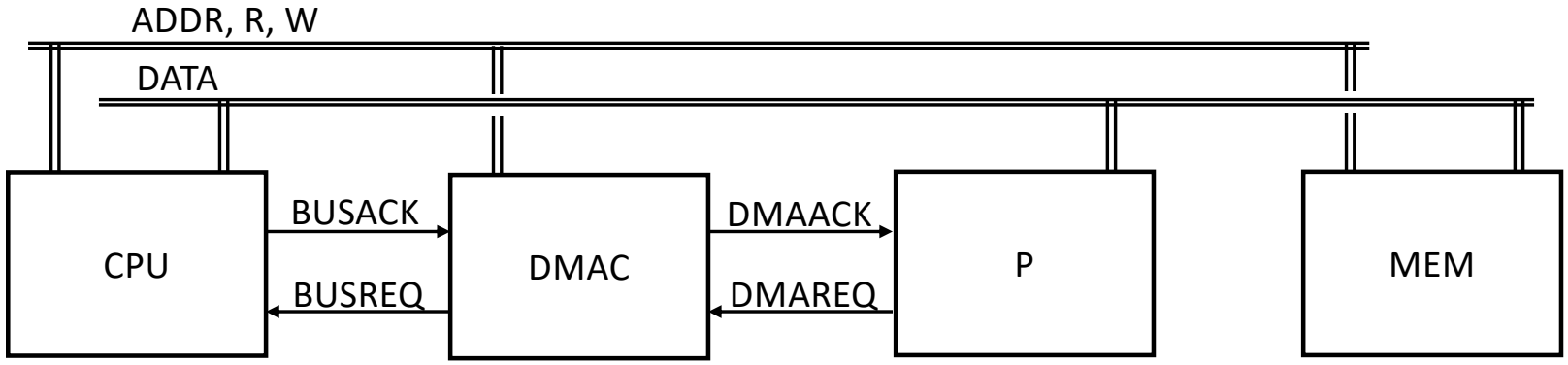
- Il tipo di gestione di periferica forse più usato, richiede un minimo di complessità hardware e software
- Dà prestazioni decenti a fronte di una non completa dedizione della MCU al trasferimento
- Utile nel caso di sistemi che non devono scambiare grandi quantità di dati con le periferiche e se non sono scambi critici in termini di latenza

Direct Memory Access (DMA)

- Il DMAC è un “tipo di periferica” utilizzata per trasferire in DMA i dati da un’altra periferica alla memoria e viceversa
- Utile nel caso di periferiche che generano grandi quantità di dati in pacchetti (es. hard disk) e che sono critiche come latenza

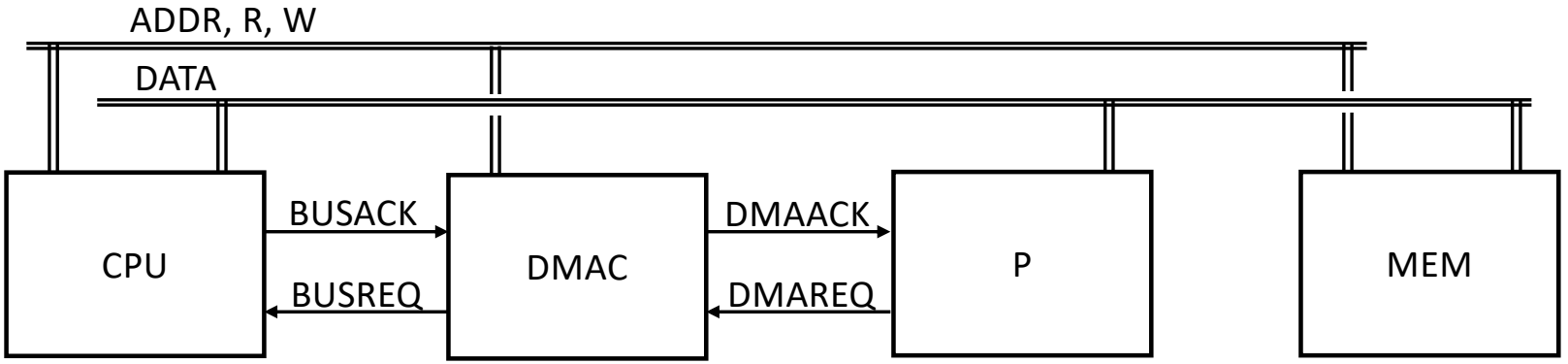
Funzionamento DMA (1)

- Supponiamo che il processore debba richiede alla periferica di effettuare, ad esempio, la lettura di un blocco di dati (trasferimento da periferica a memoria)
  - Il processore imposta il DMAC con l'indirizzo di destinazione dei dati e la quantità di dati
  - il processore fa partire il trasferimento agendo su un bit di controllo del DMAC
  - quando un dato diventa disponibile la periferica segnala al DMAC di iniziare il trasferimento (DMAREQ)



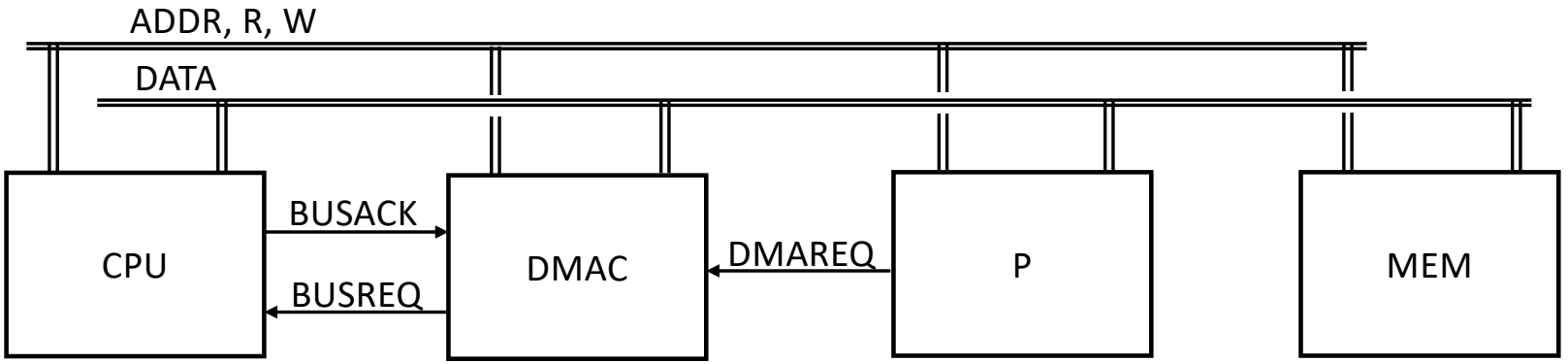
Funzionamento DMA (2)

- Il DMAC chiede al processore di sospendere l'uso del bus (BUSREQ)
- Una volta ricevuto BUSACK dalla CPU (o dal bus arbiter, se esiste) il DMAC asserisce l'indirizzo destinazione ed invia DMAACK
- La periferica pone il dato sul bus dati ed il DMAC asserisce W



Funzionamento DMA (3)

- Alternativa: Dual address model
  - Come sopra fino a ricevimento BUSACK
  - Quindi il DMAC legge il byte dal registro dati di P in un suo registro, e poi lo scrive in MEM



Funzionamento DMA (4)

- Trasferimento multi-byte:
  - burst (in blocco),
  - cycle-stealing (un byte alla volta),
  - transparent (quando il processore non usa il bus)

Timer

- Periferica che genera un segnale dopo un certo intervallo temporale
- Funzionamento:
  - Un segnale di clock viene generato dal timer stesso oppure ricevuto in input su un opportuno pin
  - Il clock può essere “scalato”, ossia passato attraverso un divisore (/ moltiplicatore) di frequenza, configurabile
  - Il clock viene utilizzato per decrementare un registro contatore ad ogni colpo; il valore iniziale del registro contatore è anch’esso configurabile
  - Quando il registro arriva a zero, viene generato il segnale di output
  - Il timer si può o meno (scelta configurabile) resettare al valore iniziale e riprendere il conteggio

Esempio: timer Intel 8253

- Il timer integrato sui PC
- Diverse modalità di funzionamento:
  - One-shot, genera un interrupt
  - One-shot, genera un impulso della durata del conteggio
  - Periodico, genera un’onda quadra con un certo duty cycle
  - Periodico, genera un’onda quadra con duty cycle = 50%
  - …

Real-time clock

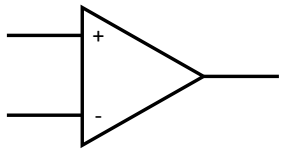
- Usato per mantenere data e ora
- Una batteria mantiene data e ora quando il sistema è spento
- Il tempo viene scandito precisamente utilizzando un oscillatore al quarzo

Watchdog timer

- Un timer con funzionamento “rovesciato”
- Invece di produrre un segnale alla scadenza di un periodo di tempo, aspetta di ricevere in input un segnale entro il tempo impostato
- Nel caso in cui tale segnale non arrivi, può essere impostata una reazione (ad es., generare un segnale usato come interrupt)
- Utilizzato per verificare periodicamente che un sistema sia “vivo”

Comparatori

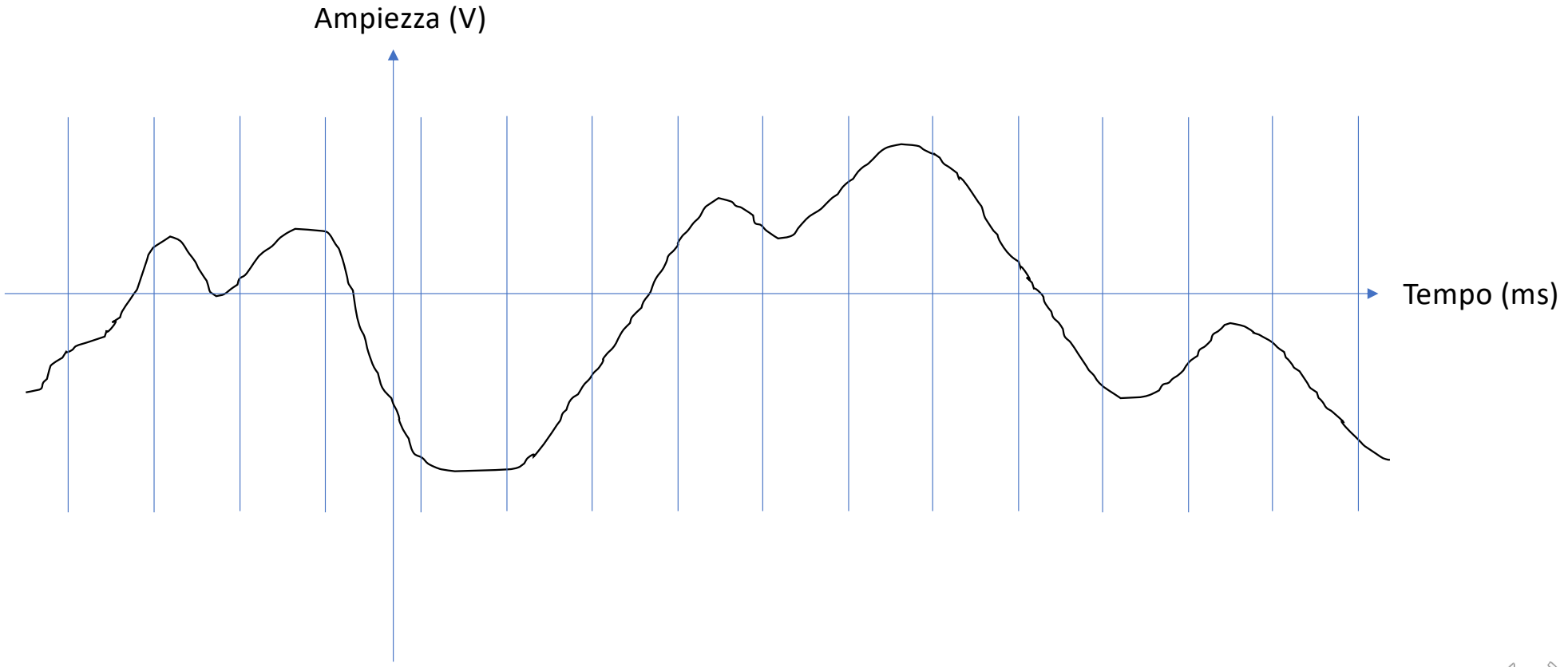
- Ricevono in ingresso due segnali analogici
- Ritornano un segnale alto o basso (0 o 1) se il primo segnale è maggiore o minore del secondo



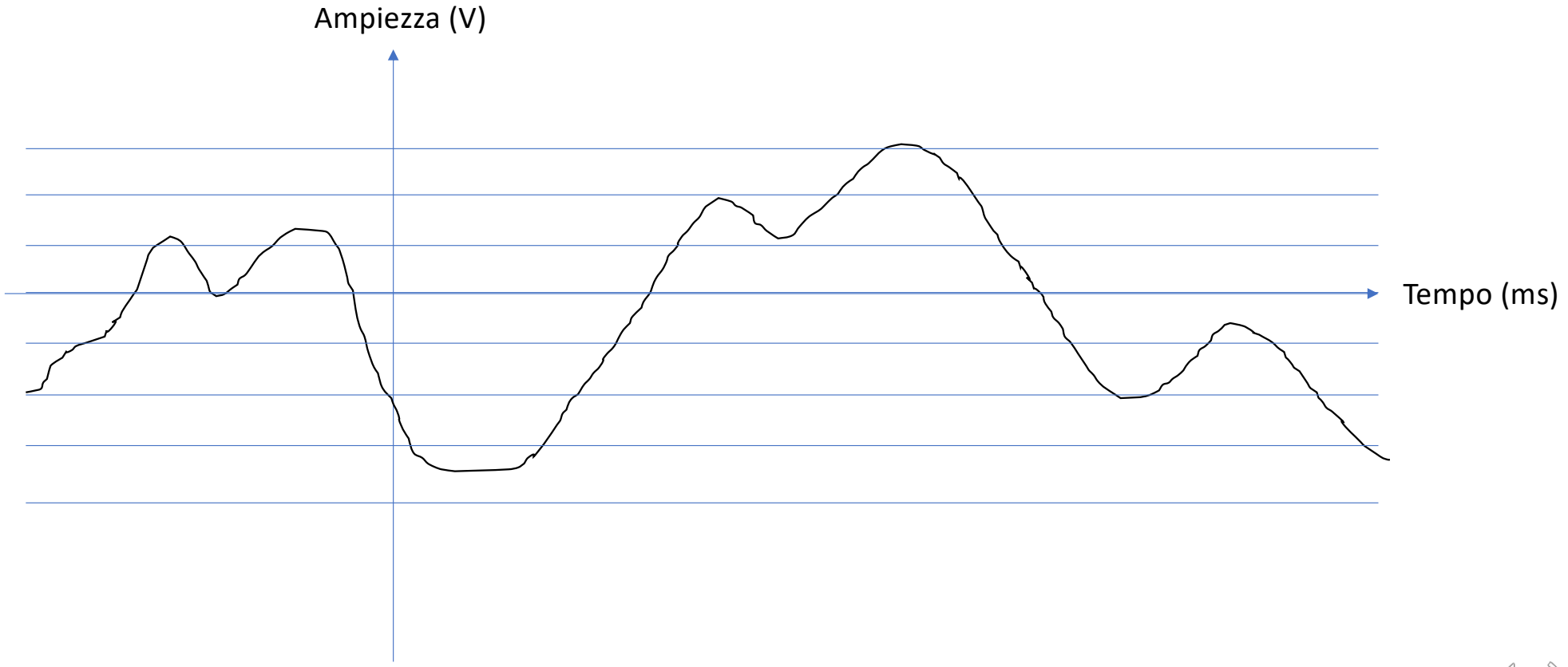
Convertitori A/D

- Periferiche che convertono un segnale (elettrico) analogico in un segnale digitale
- Due fasi:
  - Campionamento / quantizzazione del dominio = il segnale analogico e continuo nel tempo viene discretizzato nel tempo, si osservano i suoi valori ad istanti spaziati di un periodo regolare (da $\mathbb{R}$ a $\mathbb{N}$)
  - Quantizzazione del codominio = il segnale campionato viene rappresentato con una sua misura intera, rispetto ad una certa unità di misura (da $\mathbb{R}$ a $\mathbb{N}$)
  - Complessivamente il segnale passa da $\mathbb{R}$ x $\mathbb{R}$ a $\mathbb{N}$ x $\mathbb{N}$

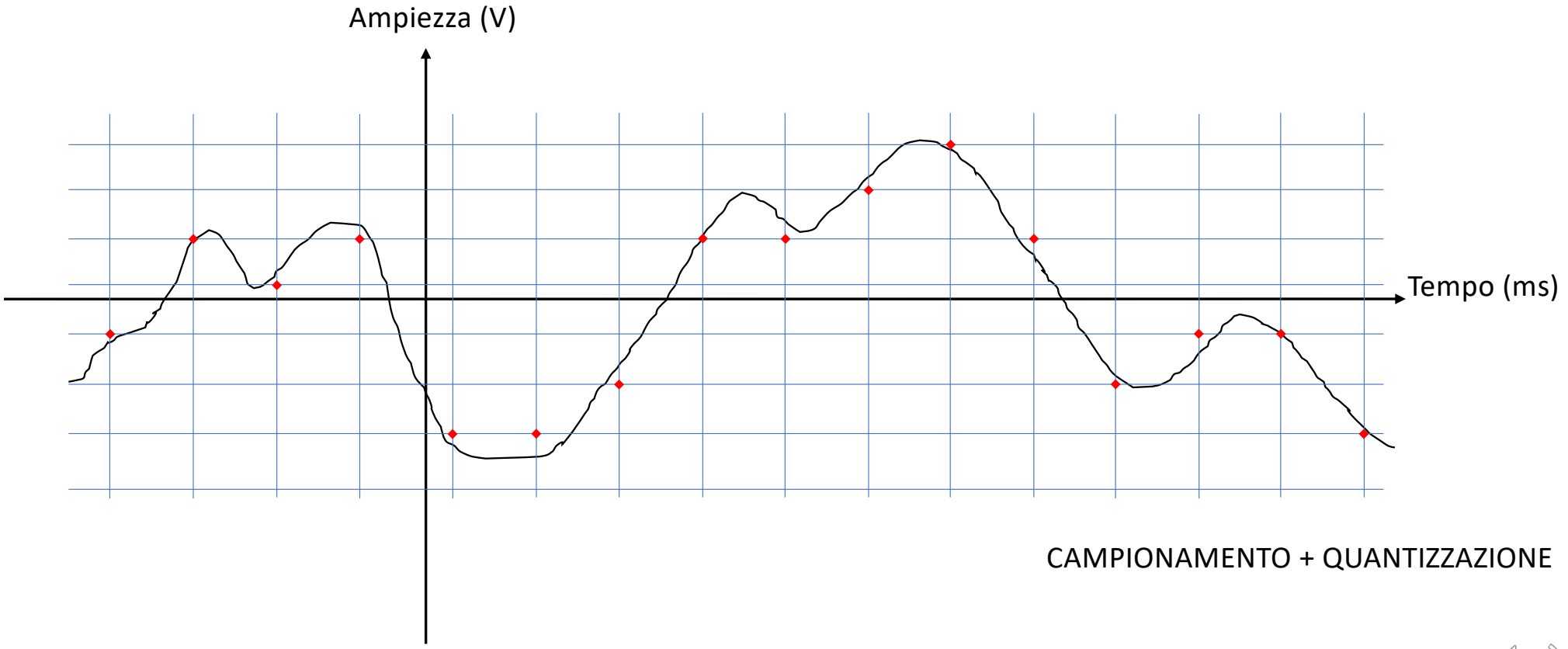
Campionamento e quantizzazione - dominio



Campionamento e quantizzazione - codominio

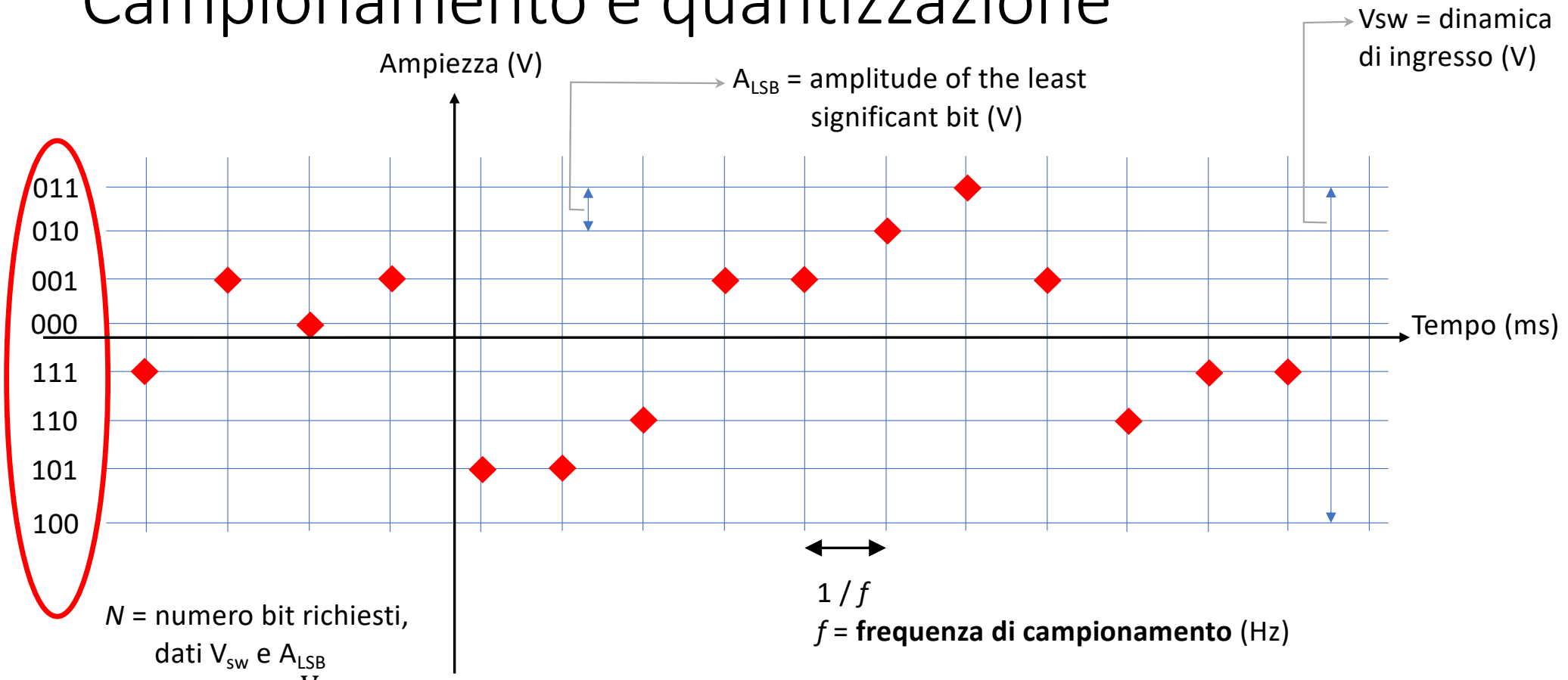


Campionamento e quantizzazione



CAMPIONAMENTO + QUANTIZZAZIONE

Campionamento e quantizzazione



N = numero bit richiesti, dati V_{sw} e A_{LSB}

$$N = \left\lceil \log_2 \left\lceil \frac{V_{sw}}{A_{LSB}} \right\rceil \right\rceil$$

Teorema del campionamento (Whittaker–Kotel'nikov–Shannon)

- Ogni segnale è ottenibile come sovrapposizione di componenti sinusoidali di frequenza e ampiezza e fase variabili
- L'intervallo di frequenze [0, f_B] in cui il segnale possiede componenti è detto la banda del segnale
- C'è una banda di ampiezza ed una banda di fase, pensiamo alle sole ampiezze (non che la fase delle componenti non sia importante!)
- Teorema del campionamento: un segnale può essere fedelmente ricostruito da un suo campionamento se la frequenza di campionamento è ≥ 2 f_B
- 2 f_B è un limite teorico (check ad esempio wikipedia oppure insegnamento Elaborazione numerica dei segnali), siccome i filtri ricostruttori reali non hanno pendenza infinita ci vuole un coefficiente ingegneristico di sicurezza (3 – 5 volte)
- Esempio: telefonia paesi occidentali, esperimento USSR, audio alta fedeltà ha una banda di circa 20 KHz; la frequenza di campionamento dei CD audio è di 44,1 KHz