



Metodo di progetto

19 maggio 2004

Politecnico di Milano

Fabio Salice & Carlo Brandolese

Sommario

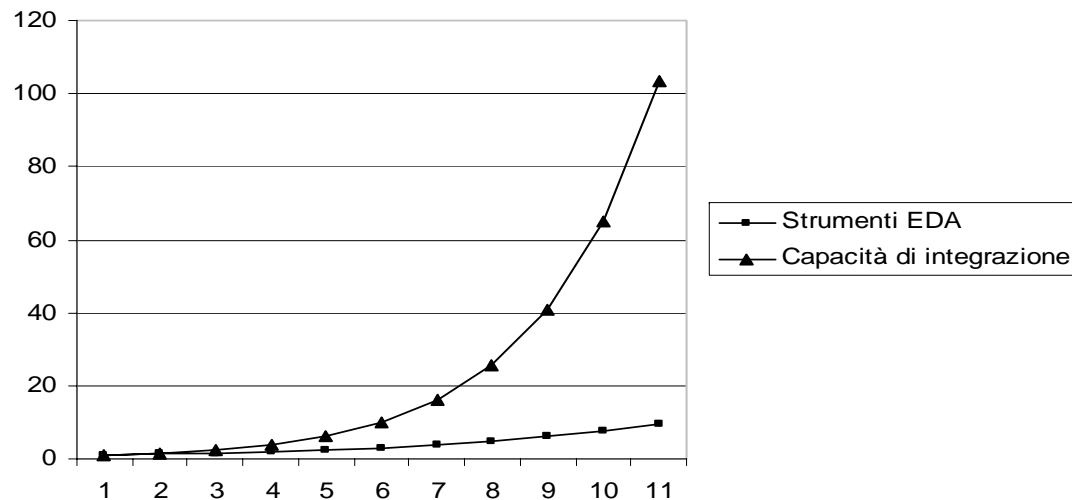


- Introduzione
 - ▶ Metodo di progetto
 - ▶ Metodo di progetto top-down e bottom-up
- Metodo di sviluppo di un progetto
- Identificazione degli aspetti del Sistema
 - ▶ Modello dell'ambiente
 - ▶ Relazione Ambiente-Sistema
 - ▶ Comunicazione tra ambiente e sistema e tra sotto-sistemi
 - ▶ Interazione tra Sotto-sistemi
 - ▶ Architettura del sotto-sistema
 - ▶ Architettura delle ALU
 - ▶ Esempio di sotto-sistema UC&DP

Metodo di progetto



- Incremento della complessità
 - ▶ Raddoppio ogni 18 mesi (legge di Moore)
- Incremento della produttività
 - ▶ Raddoppio ogni 36 mesi



- Il divario produttivo è del 28% all'anno

Metodo di progetto

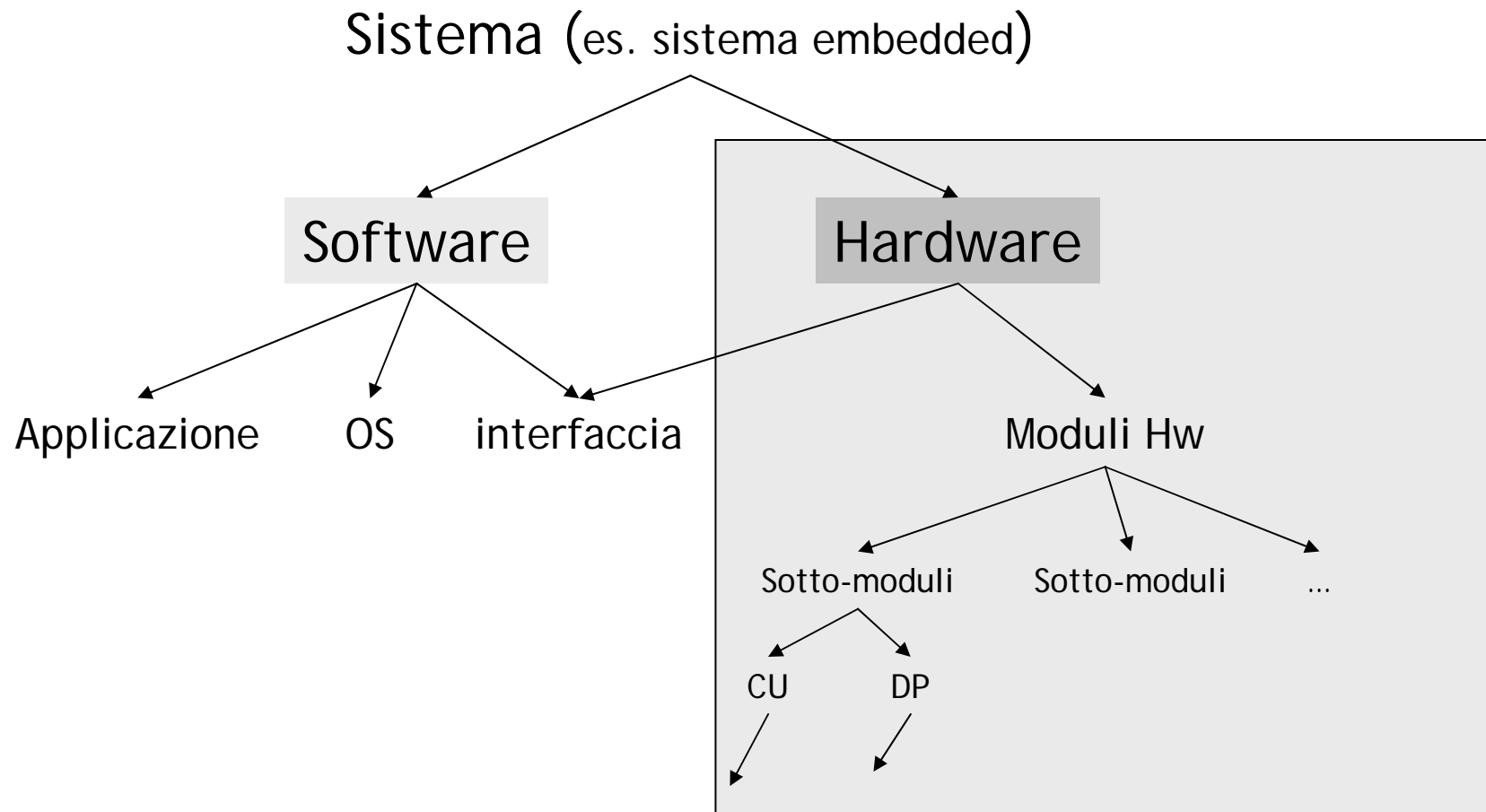


- Ridurre il divario produttivo significa gestire la complessità
- Gestione della complessità
 - ▶ Aumento del livello d'astrazione
 - ▶ Scomposizione gerarchica dell'hw
 - ▶ Strumenti EDA per l'implementazione e la sintesi, l'analisi e la verifica, la testabilità
 - Hw/Sw codesign: riduce il time to market consentendo lo sviluppo parallelo delle sezioni hw, sw e di interfaccia
 - ▶ Semplificazione
 - Modelli sincroni, stili di progetto fissati
 - IP (Intellectual properties)

Metodo di progetto



- Partizionamento gerarchico



Metodo di progetto



- L'attività di progetto è un processo creativo
 - ▶ Poche regole
 - ▶ Molta intuizione ed esperienza
- Approcci alla progettazione
 - ▶ Metodo Top-down
 - Paradigma "Divide et Impera"
 - ▶ Metodo Bottom-up
 - Composizione modulare
 - Inattuabile per sistemi di ampie dimensioni ma estremamente comodo per sistemi di dimensioni contenute

Metodo di Progetto Top-Down



- Scelta di un Algoritmo (ottimizzazione)
 - ▶ Descrizione della funzionalità da sviluppare
 - Linguaggio di specifica
 - ▶ identificazione dei moduli funzionalmente indipendenti
- Scelta di una architettura (ottimizzazione)
 - ▶ Relazione tra architettura e caratteristiche dell'applicazione
- Identificazione della gerarchia di progetto
 - ▶ Scomposizione iterativa in sotto blocchi
- Definizione delle unità base (soluzioni effettive)
 - ▶ Sommatore, macchine a stati, multiplexer, decoder, ...

Metodo di Progetto Bottom-Up



- Costruzione dei moduli generici
 - ▶ Mux, demux, priority encoder, ...
- Realizzazione della gerarchia riunendo i blocchi funzionali di base
 - ▶ Processo iterativo che conduce a moduli complessi
- Composizione dei moduli complessi per realizzare i moduli funzionali
 - ▶ I moduli funzionali sono parti della specifica
- Composizione dei moduli funzionali

Sistema

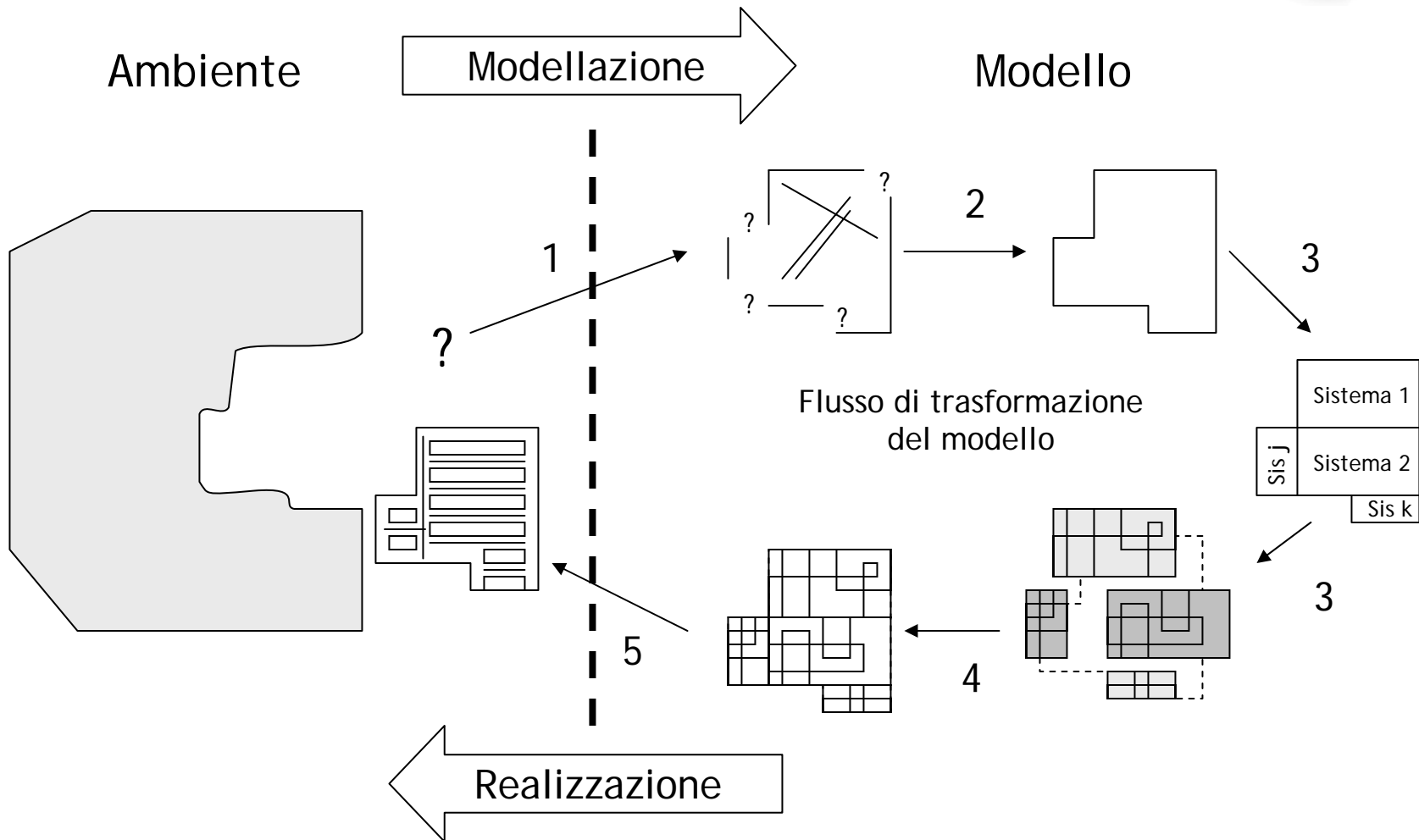


- Modello del sistema

- ▶ Modello:

- Semplificazione di un'altra realtà che può essere una realtà fisica o un altro modello
- Contiene esattamente quelle caratteristiche e proprietà della entità modellata che sono rilevanti per il compito in questione
 - Il modello è *minimale* se contiene solo gli aspetti che sono rilevanti per l'attività in questione
- Importante: non ha senso parlare di adeguatezza del modello alla realtà che intende rappresentare; la validità delle diverse descrizioni dipende dallo scopo per il quale serve la descrizione.

Metodo di sviluppo di un progetto



Metodo di sviluppo di un progetto



- Processo di Sviluppo del Sistema
 1. Raccolta dei requisiti
 - Descrizione completa e informale
 2. Raffinamento della specifica
 - Identificazione e rimozione delle ambiguità
 - Eliminazione delle ridondanze
 - Identificazione/derivazione dei vincoli
 - Vincoli:
 - » Frequenza, throughput, latenza, area, energia.
 - Interfacce con l'ambiente.
 - Interfacce: necessarie per omogeneizzare ambienti disomogenei
 - » Aspetti fisico (trasduttori, attuatori), geometrico, formale (conversione analogico digitale, codici -hamming, gray...) e temporale (asincronicità)

Metodo di sviluppo di un progetto



3. Formalizzazione della Specifica (modello)
 - Scomposizione top-down
 - Stesura della descrizione in HDL, per realizzazioni puramente hardware
 - o di sistema (SystemC, SystemVerilog, Esterel, UML, ...) nel caso di realizzazioni miste. Richiede una fase di partizionamento
4. Flusso di sintesi e operazioni di verifica
 - La verifica richiede la modellazione di parte dell'ambiente
5. Messa in opera

Metodo di sviluppo di un progetto



- Osservazione:
 - ▶ Il paradigma indicato è assimilabile alla realizzazione secondo un modello a cascata.
 - ▶ Sono possibili altri paradigmi di sviluppo dell'hw/sw
 - Prototipazione rapida
 - Modello incrementale
 - Spirale
 - Sono ereditati dall'ambito puramente sw con alcune limitazioni/variazioni dovute agli aspetti tecnologici
 - ▶ I paradigmi che richiedono lo sviluppo di versioni preliminari del dispositivo poggiano, per le fasi intermedie, su tecnologie poco costose.
 - Problemi per sistemi critici nel tempo.

Identificazione degli aspetti del Sistema

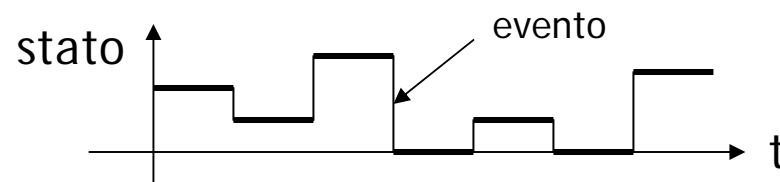


- Modello dell'ambiente
 - ▶ Adattamento del sistema alle caratteristiche significative (per il problema in esame) dell'ambiente
 - Es.: il campionamento è fatto dal sistema ma sono le caratteristiche osservate sull'ambiente a dominarne il valore
- Relazione Ambiente-Sistema
 - ▶ Interazione tra il sistema e ambiente
 - Sistema Reattivo e/o Trasformazionale
 - Sistema Statico o Adattativo
- Comunicazione tra ambiente e sistema e tra sotto-sistemi
- Architettura del sotto-sistema
- Architettura delle ALU

Modello dell'Ambiente



- Comportamento dell'ambiente visto dal sistema
 - ▶ Per il sistema, l'ambiente è discreto sia nello stato sia nel tempo
 - Stato
 - Definizione: Lo stato al tempo t_0 è l'informazione che rende univocamente determina l'uscita del sistema per $t > t_0$ quando si conosce la sequenza degli ingressi per $t > t_0$
 - Evento
 - Definizione: fenomeno associato ad una istanza di tempo e che non ha durata (punto dello spazio-tempo)
 - Stato ed evento sono correlati
 - Un evento è associato ad un cambiamento di stato.



Modello dell'Ambiente



- Quantizzazione dello stato e del tempo
 - ▶ Quantizzazione dello stato
 - Dati e Grandezze Fisiche
 - ▶ Dati
 - Numero fissato di bit, spesso definito da uno standard
 - ▶ Grandezze fisiche
 - Il numero dei livelli di quantizzazione impatta sul valore di RNS (rapporto segnale rumore), sul costo, prestazione e potenze del sistema
 - Il RNS è un valore che quantifica il rumore introdotto dalla operazione di quantizzazione
 - Tanto più elevato è il RNS tanto minore è il rumore introdotto

Relazione Ambiente-Sistema



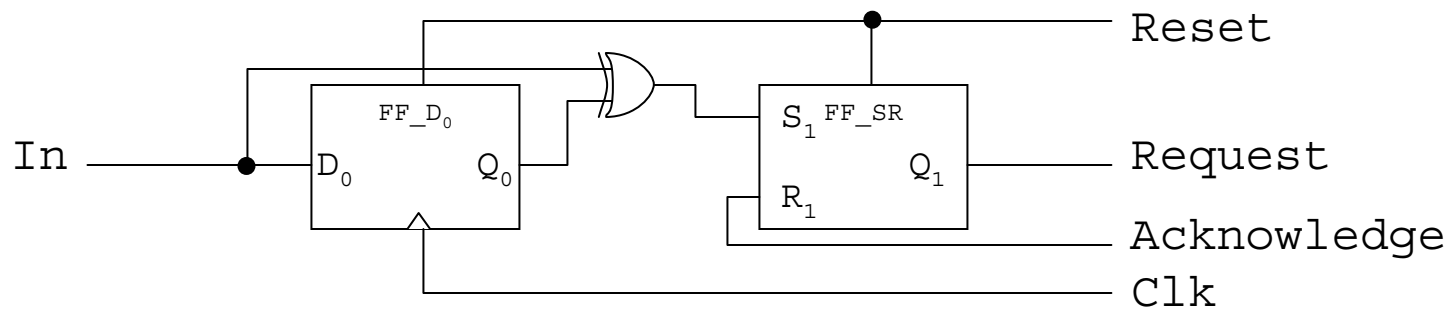
- Ruolo del (sotto)sistema indotto dall'ambiente
 - ▶ Reattività e/o Trasformazionalità
 - Reattività (o guidato dagli eventi)
 - Il sistema reagisce in tempo reale ad eventi esterni
 - » L'ambiente ha un ruolo attivo
 - Trasformazionalità
 - Campionano lo stato e restituiscono in valore
 - » L'ambiente ha un ruolo passivo
 - ▶ Osservazione:
 - I sistemi digitali sono, per natura, trasformativi.
 - Un sistema digitale può essere reso reattivo attraverso l'identificazione dell'evento
 - Differenza tra stati

Relazione Ambiente-Sistema

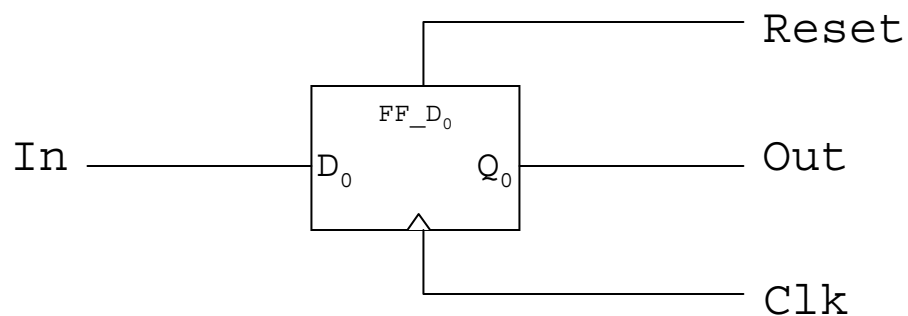


- Caratteristiche delle interfacce

- ▶ Reattività: identificazione degli eventi



- ▶ Trasformazionalità: lettura dello stato



Relazione Ambiente-Sistema

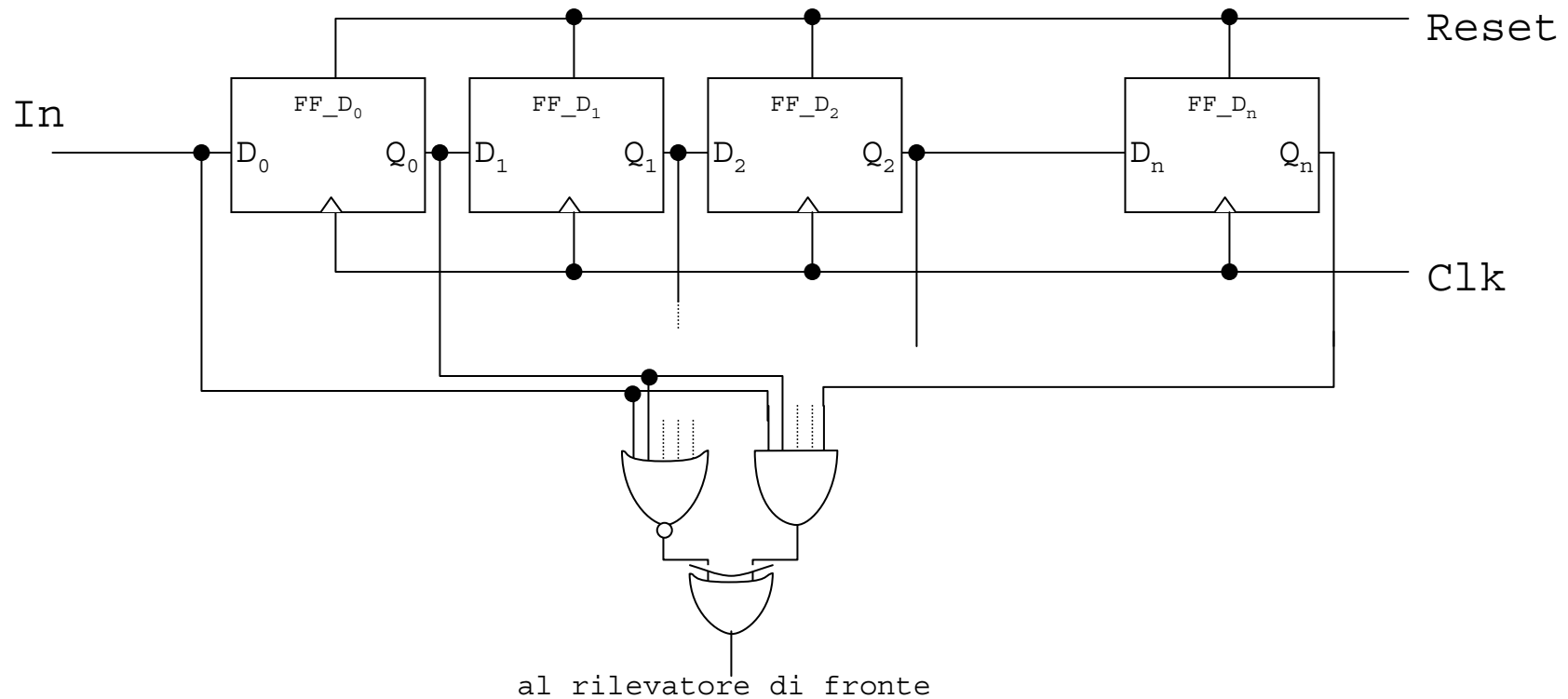


- Identificazione degli eventi
 - ▶ Necessario filtrare segnali spuri
 - Rilievo di fine transitorio
 - Accetta il nuovo stato come significativo solo dopo che permane immutato per un periodo T (transitorio)
- Lettura dello stato
 - ▶ Necessario evitare letture anomale e ridurre il rumore
 - Eliminazione delle anomalie di lettura: utilizzo del codice di Gray per la codifica dello stato dell'ambiente
 - Riduzione del rumore: sovra-campionamenti
 - ▶ Potrebbe essere richiesto il rilievo a fine transitorio

Relazione Ambiente-Sistema



- Rilievo di fine transitorio
 - ▶ La durata a livello deve essere maggiore di n ; in caso contrario non viene rilevato (considerata spuria)



Relazione Ambiente-Sistema



- Rilevazione di grandezze fisiche derivate
 - ▶ Richiede l'elaborazione di grandezze dirette
 - ▶ Es: velocità
 - Conteggio di impulsi in un periodo di tempo costante
 - Conteggio del tempo su di un numero fissato di eventi
 - ▶ Importante il dimensionamento per evitare overflow
 - Es: sotto dimensionamento dei contatori
- Generazione di segnali di controllo
 - ▶ PWM (Pulse Width Modulation): dispositivi sensibili al valor medio
 - ▶ PFM (Pulse Frequency Modulation) dispositivi sensibili alla frequenza
 - Motori passo-passo

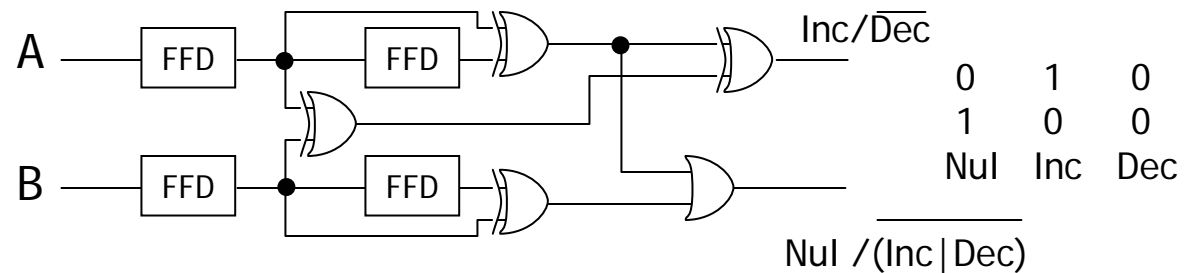
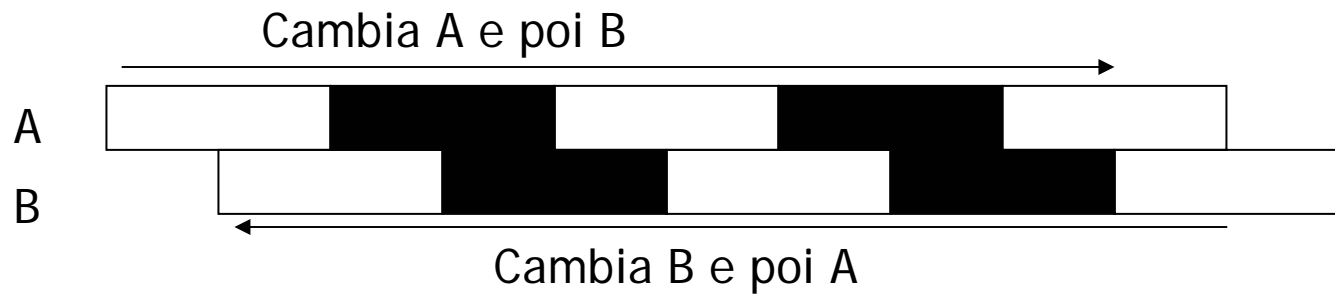
Relazione Ambiente-Sistema



- Esempio di rilevazione
 - ▶ Encoder incrementale: Rilevazione di direzione e posizione (es: mouse)

```

se (cambiamenti)
  se (cambia B)
    se (A=B) INC
    else DEC
  else
    se (A=B) DEC
    else INC
    
```



Relazione Ambiente-Sistema



- Comportamento desiderato dal (sotto)sistema
 - ▶ Statico o adattativo
 - Statico (modello tempo invariate)
 - La funzionalità del sistema non dipende dal tempo
 - Adattativo (modello tempo variante)
 - La funzionalità del sistema dipende dal tempo
 - » Es. cambiano i parametri su cui agisce un algoritmo
 - Deve essere garantita flessibilità.
 - » Utilizzo di registri per conservare i parametri;
 - » Sono raramente accettabili soluzioni che prevedono l'uso di diverse soluzioni a funzionalità intrinseca tra cui scegliere dinamicamente (soluzione semplice ma poco flessibile)

Comunicazione



- Comunicazione tra sotto-sistemi e/o tra sistema e ambiente
 - ▶ Parallelo, Seriale
 - Parallelo:
 - Costoso
 - » Buono per distanze brevi
 - Alta banda passante
 - Seriale:
 - Poco costoso
 - » ottimo per connessioni lunghe
 - Banda passante ridotta

Comunicazione



- Comunicazione tra sotto-sistemi e/o tra sistema e ambiente
 - ▶ Sincrono, Asincrono
 - Sincrono:
 - Distanze brevi
 - Problema di distribuzione del clock (skew)
 - Asincrono:
 - Distanze medie/lunghe (relativamente al sistema)
 - » Es: Systems on a Chip
 - Problema dell'overhead dovuto alla sincronizzazione della trasmissione
 - » Protocollo
 - ▶ protocollo di comunicazione
 - Definisce le regole di sincronizzazione ed il formato dei dati

Comunicazione



- Esempi di protocollo (semplici)
 - ▶ Modalità di sincronizzazione
 - Sincrona breve distanza: Clock
 - Asincrona breve distanza: Fully-Interlock (Ready - Accept)
 - Sincrona lunga distanza: codici di autosincronizzazione
 - Manchester (autosincronizzante): transizione da 0 a 1 (per un 1) e da 1 a 0 (per uno 0) a metà periodo. Esiste almeno una transizione per periodo (indipendentemente dal messaggio)
 - Asincrona lunga distanza: Bit di Start e Stop
 - ▶ Codifica
 - Parità (pari o dispari)
 - CRC

Comunicazione



- Interazione tra sottosistemi
 - ▶ Cooperazione
 - Interazione desiderata a prevista
 - ▶ Conflitto
 - Interazione non desiderata e non prevista
 - ▶ Interferenza
 - Interazione non desiderata ma prevista
- Conflitto ed interferenza nascono quando sono condivise delle risorse
 - ▶ Es: Comunicazione su BUS (Parallelo, Sincrono)
- Sottosistemi cooperanti sono indipendenti, non condividono risorse, e si scambiano dati in modo sincrono e previsto
 - ▶ Il controllo tra sottosistemi è inesistente

Comunicazione



- Sottosistemi che condividono delle risorse richiedono un controllo esplicito per risolvere il problema del conflitto e della interferenza
- Due possibili paradigmi di controllo
 - ▶ Concentrato:
 - Architetture Master-Slave
 - Le unità Slave inoltrano la richiesta che viene soddisfatta dal Master seguendo una qualche politica
 - ▶ Distribuito:
 - La decisione è derivata dai sottosistemi cooperanti
 - Tipicamente si assegna una priorità alle unità
 - » Cablata
 - » programmata

Architettura di un Sotto-sistema



- I sottosistemi possono avere una architettura
 - ▶ Diretta al Controllo
 - Macchina a stati
 - ▶ Diretta alla Elaborazione
 - Collezione di ALU
 - ▶ Ibrida
 - Unità di controllo concentrata e Data-Path concentrata
 - Controllo e Data-Path distribuiti
 - Non si riconosce più il controllo dal resto

Architettura di un Sotto-sistema esempio



- Specifica di una serratura a combinazione
 - ▶ Fornire 3 valori in sequenza per aprire una porta;
 - ▶ Un errore lascia la serratura chiusa e richiede di ripartire; Il risultato è noto solo ad inserimento completato.
 - ▶ Alla chiusura della porta, il dispositivo viene riattivato e si pone in attesa di una nuova sequenza
 - ▶ Ingressi:
 - Valore in ingresso che appartiene alla sequenza;
 - Reset (per inizializzare il sistema);
 - ▶ Uscite:
 - porta aperta/chiusa;

Architettura di un Sotto-sistema esempio

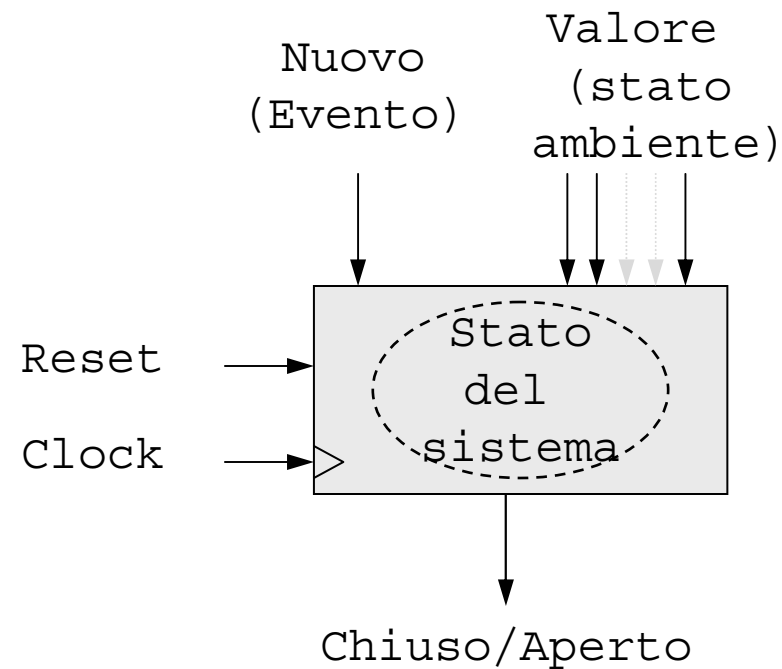


- Aspetti non definiti e non impliciti:
 - ▶ Il numero bit per la codifica sia del valore in ingresso sia dell'uscita (stato dell'ambiente);
 - Si specifica che le cifre lette sono 3 ma non se ne definisce l'intervallo d'appartenenza.
 - Non è implicito che la tastiera sia costituita dalle 10 cifre 0:9.
- Aspetti impliciti da evidenziare:
 - ▶ Identificazione di un cambiamento di stato dell'ambiente
 - Comportamento reattivo
 - ▶ Il sistema è sequenziale
 - È necessario fornire una sequenza di valori ricordando la storia degli ingressi;
 - È necessario ricordare se si è verificato un errore;

Architettura di un Sotto-sistema esempio



- Rappresentazione logica del dispositivo
 - ▶ Pin-out logico



Architettura di un Sotto-sistema esempio

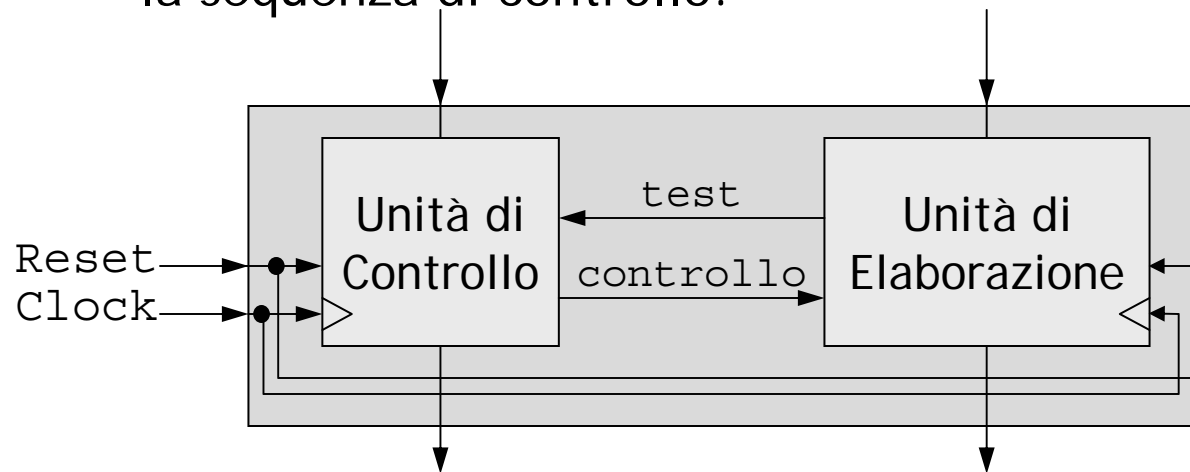


- Due possibili architetture del sistema
 - ▶ Come unica macchina a stati
 - Ogni evento e stato (ambiente) porta il sistema in un nuovo stato: valore esatto o valore errato.
 - Due possibilità
 1. FSM cablata sulla sequenza da riconoscere
 2. FSM generica con rete di transcodifica per la normalizzazione degli ingressi
 - » la struttura è più flessibile; una modifica della sequenza da riconoscere incide solo sulla progettazione della rete di transcodifica.

Architettura di un Sotto-sistema esempio



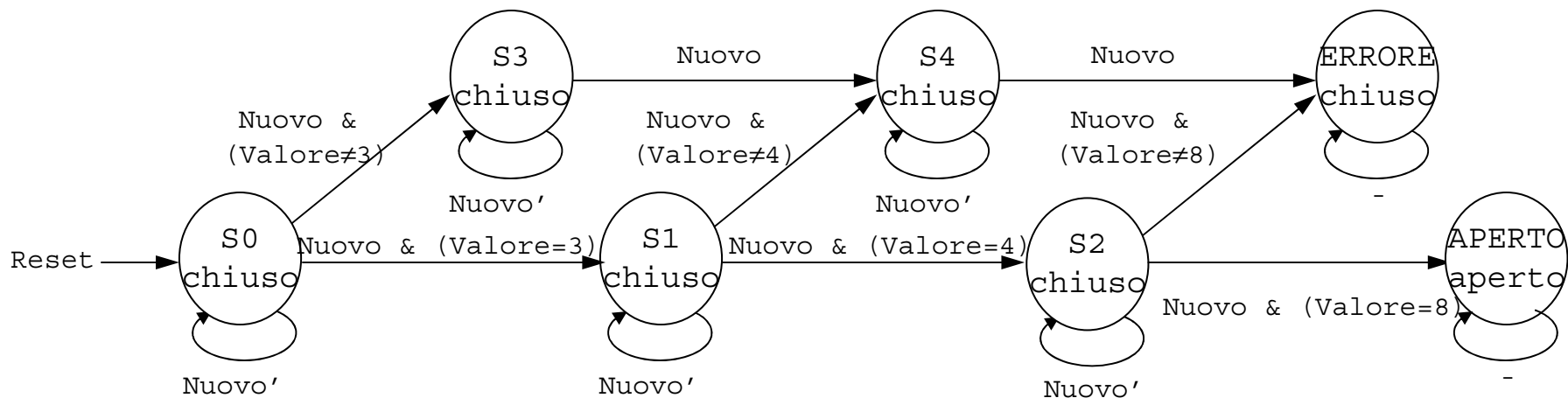
- Due possibili architetture del sistema(Cont.)
 - ▶ Come Unità di Controllo e Unità di Elaborazione
 - L'unità di controllo è realizzata come macchina a stati; risulta generale ed indipendente dalla sequenza da riconoscere. Comanda l'unità di elaborazione.
 - L'unità di elaborazione controlla i valori immessi e restituisce alla unità di controllo un valore che modifica la sequenza di controllo.



Architettura di un Sotto-sistema esempio



- Macchina a stati 1
 - ▶ Sviluppo con i criteri noti
 - Ipotesi: i valori da riconoscere sono 3,4,8
 - 4 bit di ingresso (minimo valore)



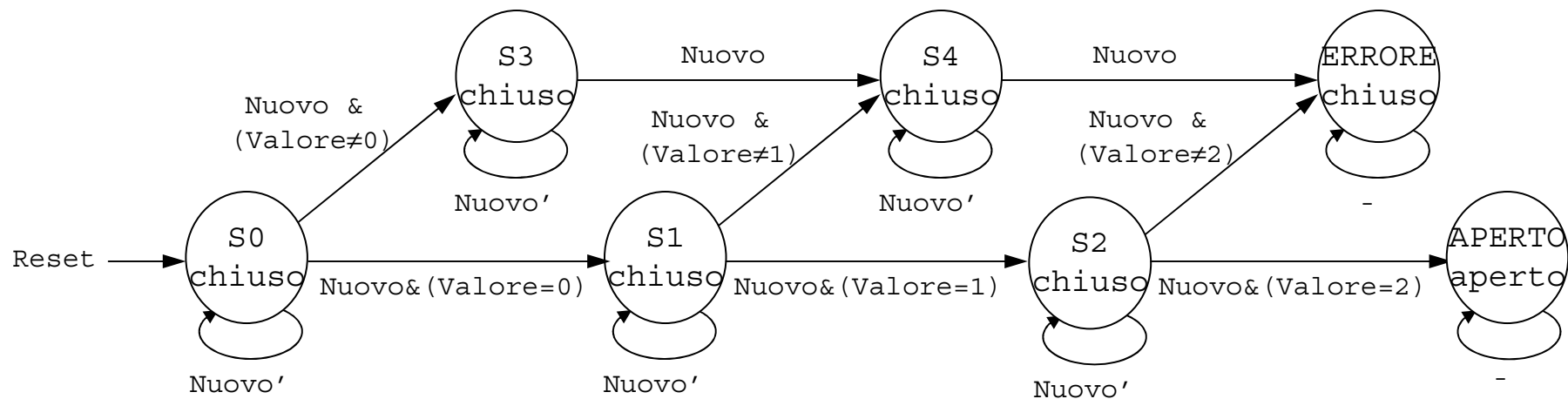
Dimensione dei segnali
Reset: 1 bit
Nuovo: 1 bit
Valore: 4 bit
Chiuso/Aperto: 2 bit

Architettura di un Sotto-sistema esempio



- Macchina a stati 2

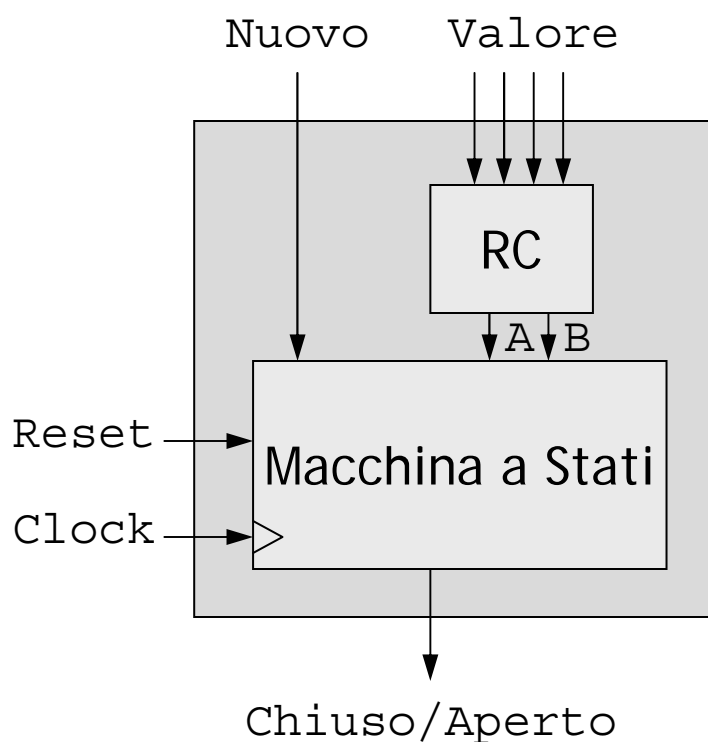
- ▶ Il funzionamento della macchina a stati è indipendente dai valori da identificare
- ▶ La rete combinatoria codifica i valori di ingresso uniformandoli a quelli per i quali la macchina a stati è progettata
 - I valori da riconoscere arbitrari (es: 0,1 e 2)



Architettura di un Sotto-sistema esempio



- Mantenendo l'ipotesi sui valori da riconoscere (3,4,8), si ottiene:



V_3	V_2	V_1	V_0	A	B
0	0	1	1	0	0
0	1	0	0	0	1
1	0	0	0	1	0

$$A = P0 * (V_3 + V_2' + V_1 + V_0)$$

$$B = P0 * (V_3' + V_2 + V_1 + V_0)$$

$$P0 = (V_3 + V_2 + V_1' + V_0')$$

Esempio di altra sequenza
(7, 0, 15)

V_3	V_2	V_1	V_0	A	B
0	1	1	1	0	0
0	0	0	0	0	1
1	1	1	1	1	0

$$A = P0 * (V_3 + V_2 + V_1 + V_0)$$

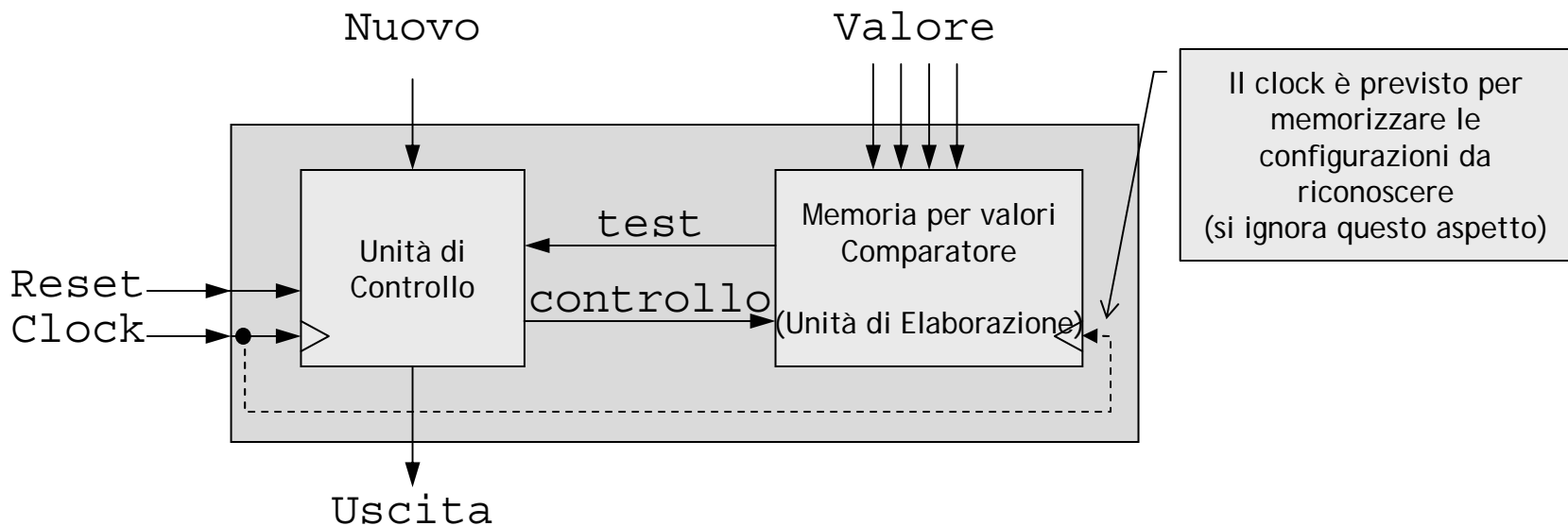
$$B = P0 * (V_3' + V_2' + V_1' + V_0')$$

$$P0 = (V_3 + V_2' + V_1' + V_0')$$

Architettura di un Sotto-sistema esempio



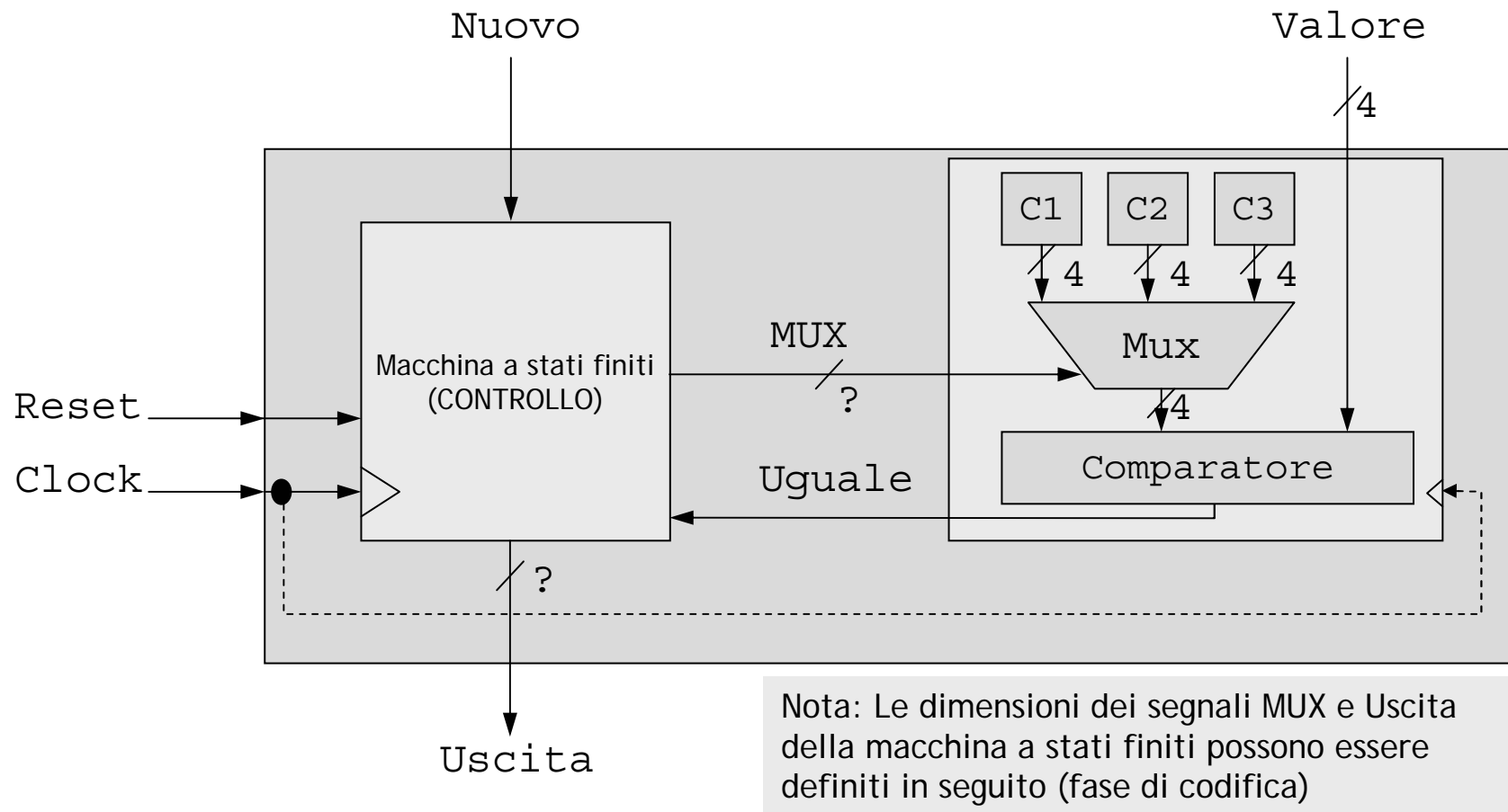
- Unità di Controllo e Unità di Elaborazione
 - ▶ Possibile realizzare una struttura a funzionalità programmabile
 - Le configurazioni sono memorizzate e vengono confrontate con i valori immessi passo dopo passo
 - Flessibilità alta; maggior complessità di realizzazione



Architettura di un Sotto-sistema esempio



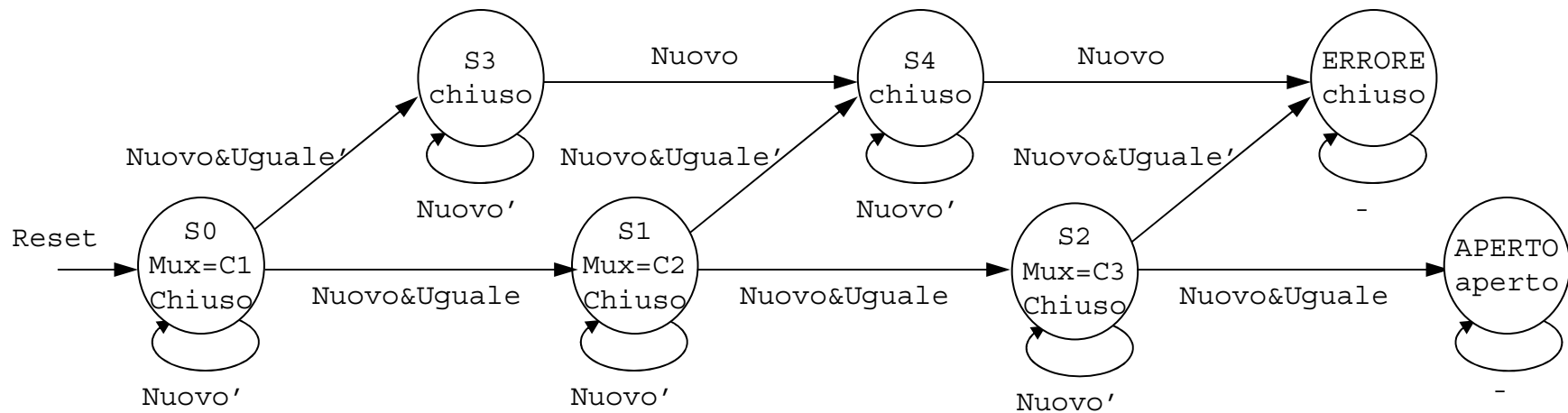
- Struttura circuitale



Architettura di un Sotto-sistema esempio



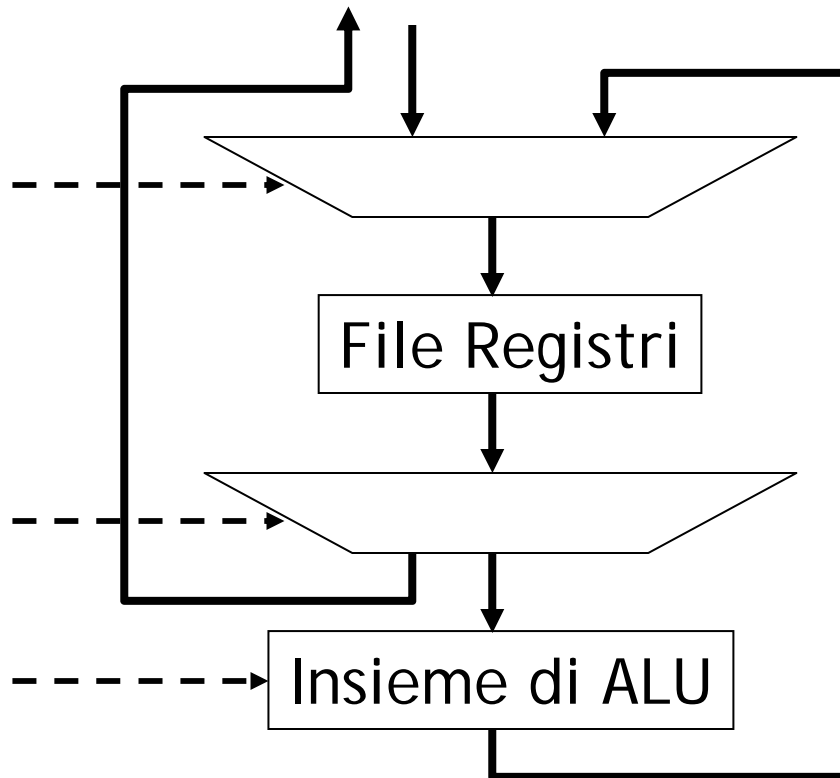
- Grafo degli stati della Unità di controllo



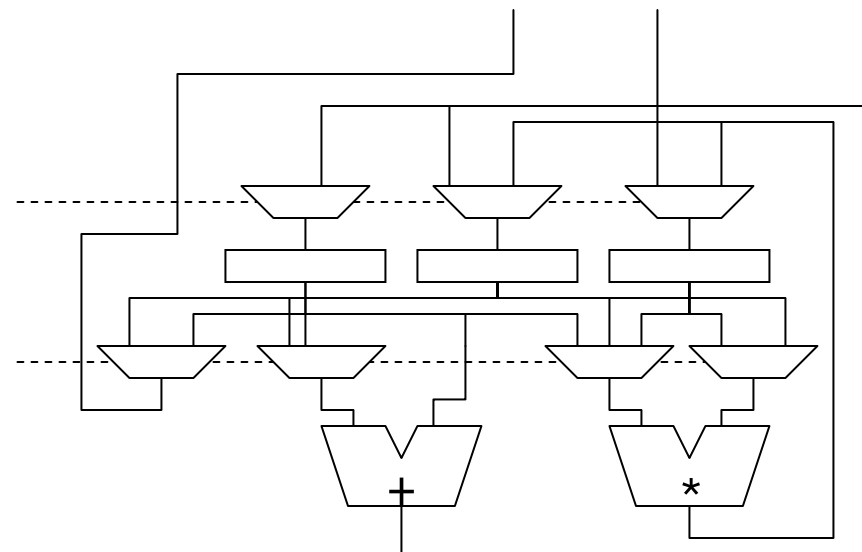
Architettura di un Sotto-sistema



- Struttura del Data-Path (basata su multiplexer)



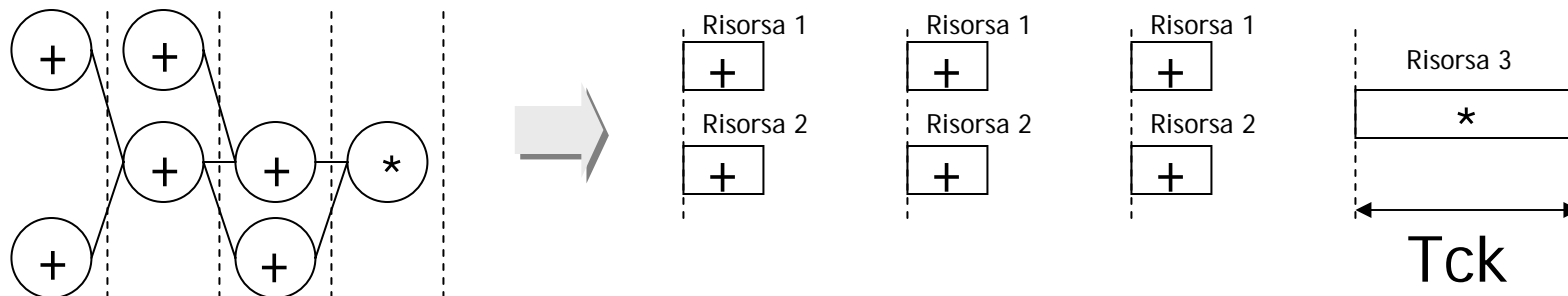
Esempio



Architettura di un Sotto-sistema



- Struttura del Data-Path (basata su multiplexer)
 - ▶ Relazione spazio-tempo:
 - un aumento del numero delle ALU allocate (risorse di calcolo) permette di ridurre la latenza
 - Una riduzione del numero delle risorse di calcolo impatta sia sulla sezione di multiplexing sia sul numero di registri
 - I due aspetti sono correlati
 - Il periodo di clock è dominato dalla risorsa di calcolo più lenta



Architettura di un Sotto-sistema

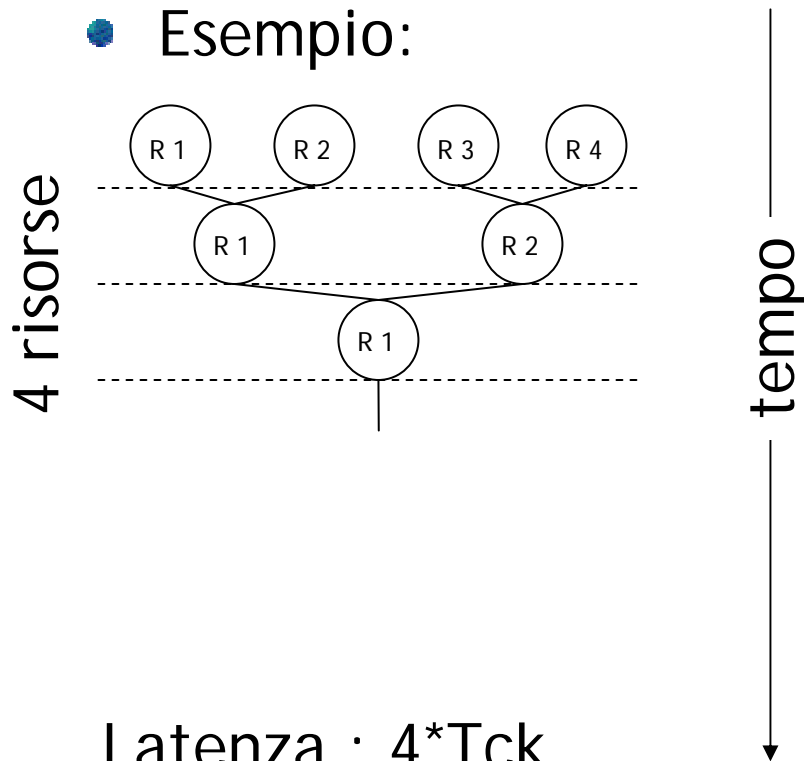


- Definizioni
 - ▶ Latenza
 - Tempo necessario per produrre il risultato di una elaborazione completa su di un insieme di dati
 - ▶ Throughput
 - Quantità di risultati prodotti per unità di tempo
 - ▶ Prestazione
 - Caratteristica del sistema che ne indica la rapidità di calcolo; in generale, è costituita dall'insieme {latenza, throughput, Tck}
 - Nota: Il periodo di clock non è sufficiente per definire le prestazioni di un sistema

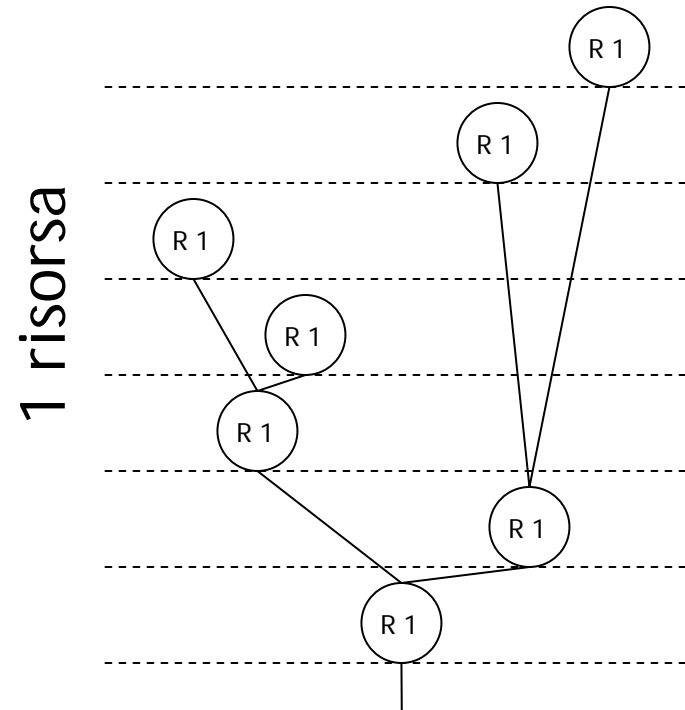
Architettura di un Sotto-sistema



● Esempio:



Latenza : $4 \cdot T_{ck}$
Throughput: $1/4$
Costo Reg: 4
Costo Mux: $3 (2-a-1)$



Latenza : $8 \cdot T_{ck}$
Throughput: $1/8$
Costo Reg: 5
Costo Mux: $5 (2-a-1)$

Architettura di un Sotto-sistema



- Aumento delle prestazioni
 - ▶ Riduzione della latenza
 - Aumento del numero di risorse di calcolo
 - Limite fissato dalla dipendenza dei dati
 - Concatenamento
 - ▶ Aumento del throughput
 - Aumento del numero di risorse di calcolo
 - Limite fissato dalla dipendenza dei dati
 - Concatenamento
 - Pipeline
 - ▶ Riduzione del periodo di clock
 - Multi-ciclo
 - Architetture di calcolo ottimizzate in prestazione
 - Es: da *Ripple Carry Adder* a *Carry Look Ahead Adder*

Esempio di progetto



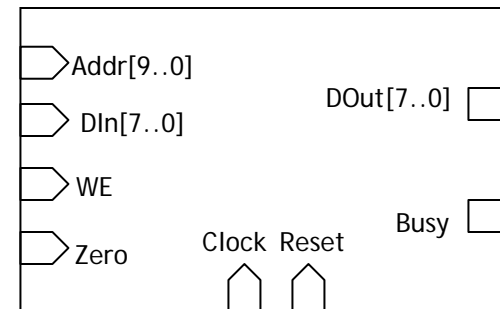
- Specifiche
 - ▶ Realizzare un blocco di controllo ad una RAM capace di portare a 0 il contenuto di un intervallo di celle.
 - ▶ L'intervallo è indicato da un indirizzo LOW e da un indirizzo HIGH
 - Tutte le celle dall'indirizzo LOW all'indirizzo HIGH, estremi inclusi, vengono azzerate
 - ▶ Un segnale di ingresso specifico (`Zero`) attiva il processo di caricamento degli indirizzi e l'attivazione della procedura di azzeramento.
 - ▶ Per tutta la fase di azzeramento viene attivato un segnale di `Busy`

Esempio di progetto



- Ingressi

- ▶ Clock e Reset
- ▶ Addr
- ▶ DataIn
- ▶ WE (Write Enable)
- ▶ Zero



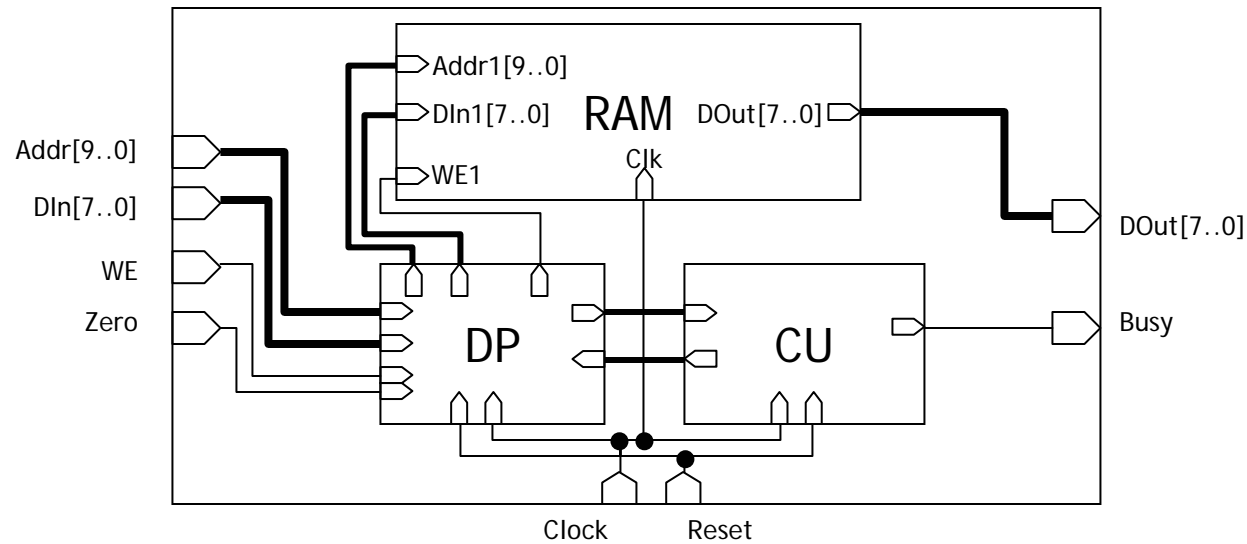
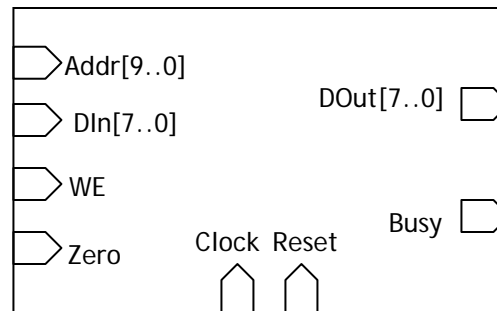
- Uscite

- ▶ DataOut
- ▶ Busy

Esempio di progetto



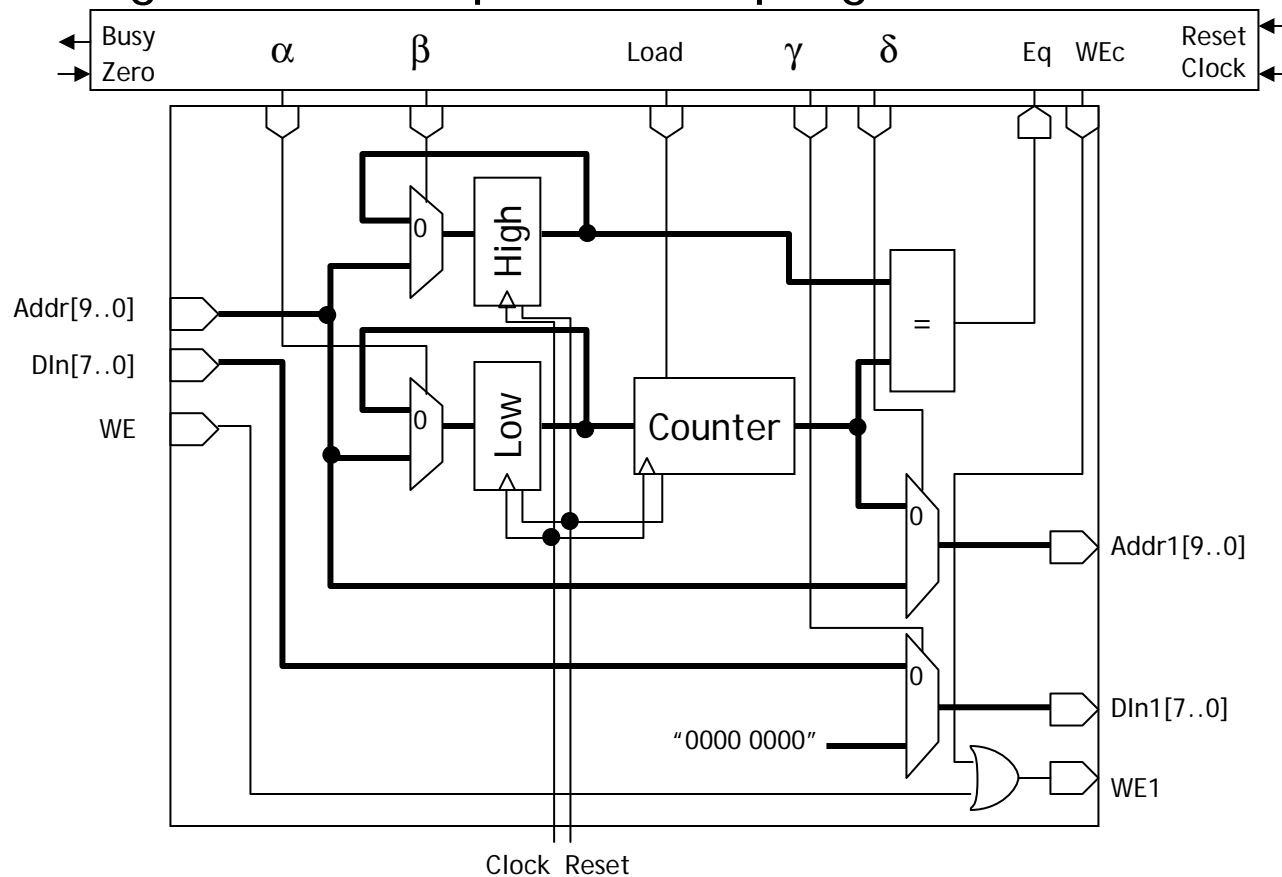
- Scomposizione Top-Down (1 livello)



Esempio di progetto



- Scomposizione Top-Down (2 livello)
 - ▶ Progetto del DP procede il progetto dalla UC



Esempio di progetto



- Scomposizione Top-Down (2 livello)

- ▶ Progetto dalla UC

Sp	Sf	Zero	Eq	α	β	γ	δ	Load	WEc	Busy	
S0	S0	0	-	0	0	0	1	-	0	0	
S0	S1	1	-	0	1	0	1	-	0	1	-- Carica il registro HIGH
S1	S2	-	-	1	0	0	1	-	0	1	-- Carica il registro LOW
S2	S3	-	-	0	0	0	1	1	0	1	-- Carica il contatore
S3	S3	-	0	0	0	1	0	0	1	1	-- Azzeramento Ciclico
S3	S0	-	1	0	0	0	1	0	0	0	

- ▶ Sommario

- Progettato il DP e individuata la UC il progetto di un sotto-sistema è sostanzialmente terminato
 - Implementare e simulare il DP per primo consente di trovare eventuali errori e corregge il progetto della UC
 - Esempio: problemi di sincronizzazione
 - Si osservi che sono state prese delle decisioni arbitrarie come, ad esempio, quale indirizzo è caricato per primo.

Bibliografia



- L. Mezzalana, "Informatica Industriale", 3^a Edizione
- E. Bava, R. Toboni, C. Svelto, "Fondamenti della misurazione", Esculapio, Bologna, 2003.
- S. Brown and J. Rose, "Architecture of FPGAs and CPLDs: A Tutorial", IEEE Design and Test of Computers, vol. 13, no. 2, 1996, pp. 42-57
- H. Wolkerstorfer, "VLSI-Design", Appunti del corso, Gratz University of Technology, 2003
- A. Jantsch, "System Modelling", Appunti del corso, Royal Institute of Technology, Stockholm, 2004