UNIVERSITÀ DI PISA

CORSO DI LAUREA IN FISICA

LABORATORIO DI FISICA 3

MACCHINE A STATI FINITI
FINITE STATES MACHINES

Prof. F. Forti

Finite states machine

- Meccanismo per permette concettualizzare e formalizzare il funzionamento di un sistema logico complesso
- Universalmente utilizzate per la realizzazione di sistemi hardware e software
- L'implementazione puo' essere realizzata in molti modi:
  - Hardware – con componenti standard
  - Hardware – con logica programmabile (FPGA)
  - Software – microcontrollori e microprocessori
  - Software – sistemi distribuiti

Finite State Machines

- Ho un insieme finito di stati S: $\{s_i\}$
- Un insieme finito di inputs I: $\{i_k\}$
- Una funzione di transizione: $S \times I \rightarrow S$
  - Specifica in qualo stato va la FSM al ciclo successivo, dato lo stato di partenza e l'input ricevuto.
  - Si puo' rappresentare con un grafo o con una matrice
- In aggiunta c'e' un insieme di outputs O: $\{o_j\}$
- Gli outputs possono essere funzione dello stato (Moore FSM) oppure della transizione (Mealy FSM)
- NB: importante strumento concettuale anche per la scrittura di software.
  - Anche generabili automaticamente. Ad es. AutoFSM

http://www.gnu.org/software/autogen/autofsm.html

Diagramma di stati

- Un diagramma di stati specifica tutti gli stati di un sistema, le transizioni tra i vari stati e le regole con cui si passa da uno stato all'altro.
- Il diagramma di stati e` la rappresentazione grafica di una astrazione matematica molto importante che si chiama Macchina a Stati Finiti (Finite State Machine).
- Il diagramma di stati aiuta nella progettazione dei circuiti logici sequenziali. Si passa dal diagramma al circuito
- Assegnando un codice binario ad ogni stato es.: per i 4 stati mostrati ho bisogno di 2 FF

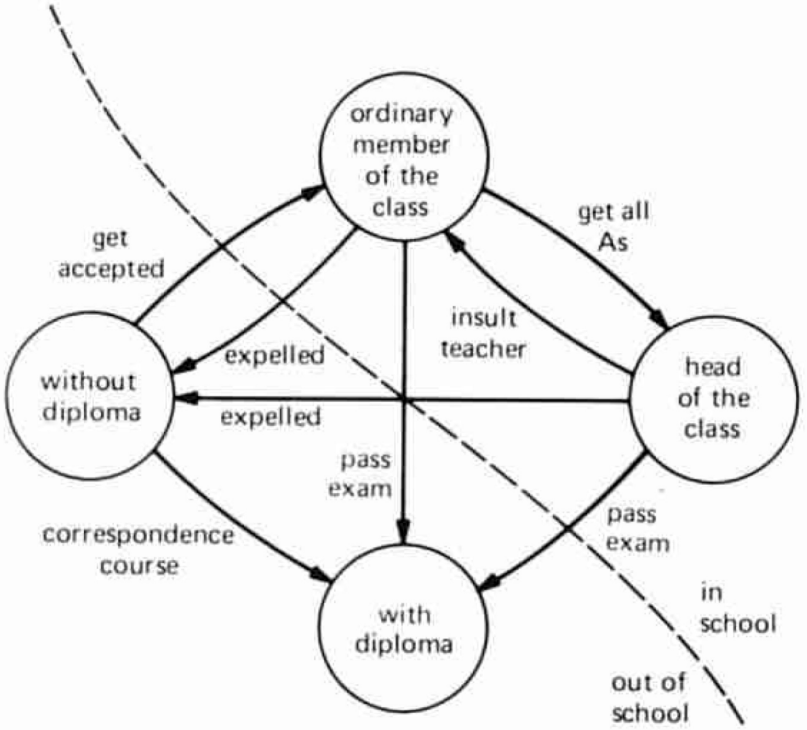


Figure 8.61. State diagram: going to school.

State Diagram: going to school
The Art of Electronics, Horowitz, Hill

Macchina a Stati Finiti - FSM

- Il diagramma di stati e` la rappresentazione grafica di una astrazione matematica che si indica con Macchina a Stati Finiti (Finite State Machine – FSM)
- La Macchina a Stati Finiti e` molto utile per la progettazione di reti logiche sequenziali
- Il contatore e` un esempio molto semplice di FSM, il suo "prossimo" Stato (valore dell'insieme delle uscite) dipende solo dal valore dello Stato "attuale"
- I contatori inoltre non hanno alcun ingresso definito da un segnale "esterno" a parte il segnale di Clock
- Il diagramma dalla vita reale che abbiamo appena visto e` invece piu` complesso, il passaggio di stato e` subordinato ad una azione esterna che possiamo vedere come un ingresso definito da un segnale esterno

FSM e diagramma di stati

I possibili STATI del sistema sono specificati nelle bolle del diagramma

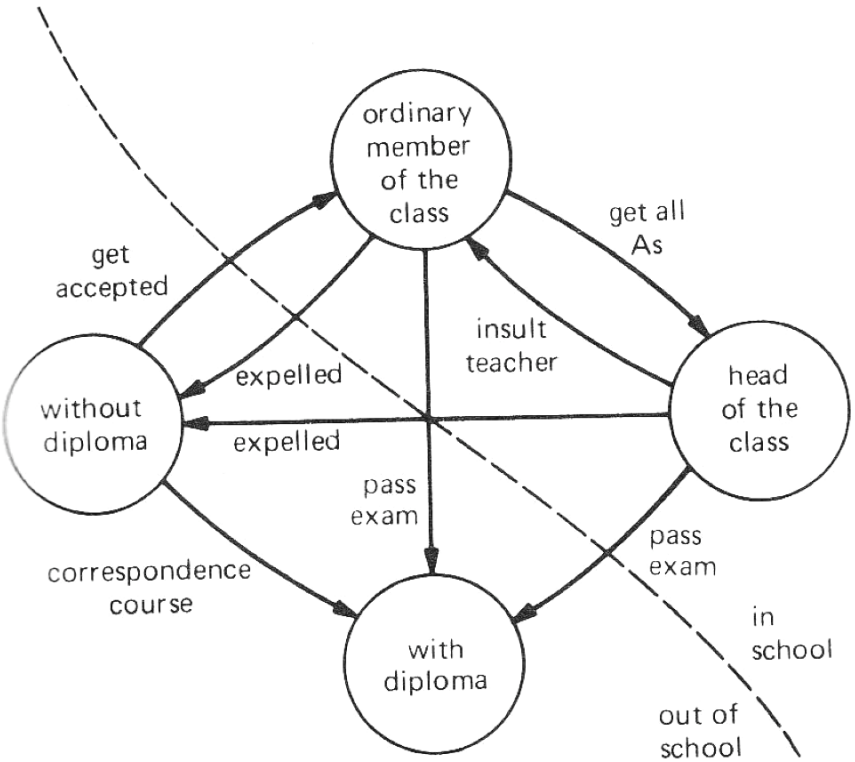
Lo STATO attuale e` definito dal valore delle uscite Q, o da una combinazione di esse

Le transizioni da uno stato all'altro sono sincronizzate con il segnale di CLOCK

Il valore dei segnali esterni che determinano le transizioni sono indicate sulle "freccie" che indicano il passaggio di stato

Il prossimo STATO e` determinato dai valori degli ingressi (D nel caso di FF D)

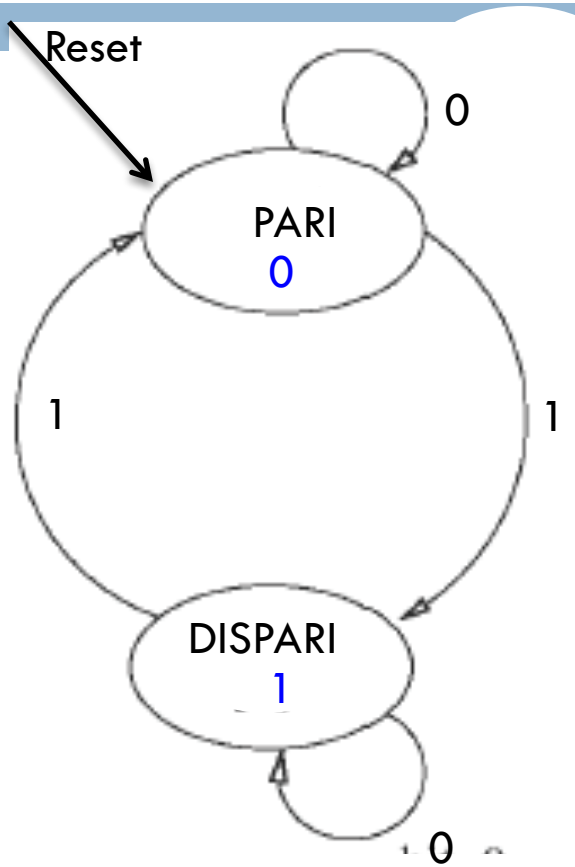
La logica combinatoria permette di definire I valori degli ingrressi = fn(Q,Segnali Esterni)



State Diagram: going to school
The Art of Electronics, Horowitz, Hill

Esempio di FSM: controllo di parita`

1. Disegnare il diagramma di stato
2. Scegliere gli stati del sistema, definiti dal valore delle uscite
3. Tabella della verita` dei valori delle uscite dello stato attuale
4. Scelta dei FF con cui voglio implementare la logica sequenziale -> valori degli ingressi per produrre i valori degli ingressi che genereranno il prossimo stato
5. Trovare la rete combinatoria corretta per produrre gli ingressi dalle uscite available. Per fare questo si puo` anche utilizzare le mappe di Karnaugh.

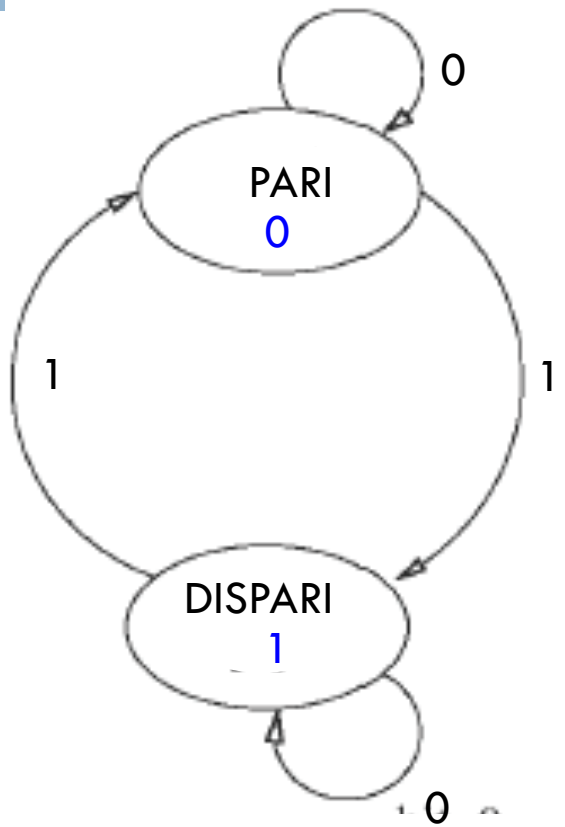


Blu = Rappresentazione dello stato

Esempio di FSM: controllo di parita`

StatoAttuale	Input	StatoFuturo
Pari/0	1	Dispari/1
Dispari/1	1	Pari/0
Pari/0	0	Pari/0
Dispari/1	0	Dispari/1

3. tabella della verita` dei valori delle uscite dello stato attuale



Blu = Rappresentazione dello stato

Esempio di FSM: controllo di parita`

StatoAttuale	Input	StatoFuturo
Pari/0	1	Dispari/1
Dispari/1	1	Pari/0
Pari/0	0	Pari/0
Dispari/1	0	Dispari/1

Scegliamo per esempio FF di tipo D.
La Tabella della verita` :
Stato Attuale == Q
Stato Futuro == D

Q	Input	Q+	D
0	1	1	1
1	1	0	0
0	0	0	0
1	0	1	1

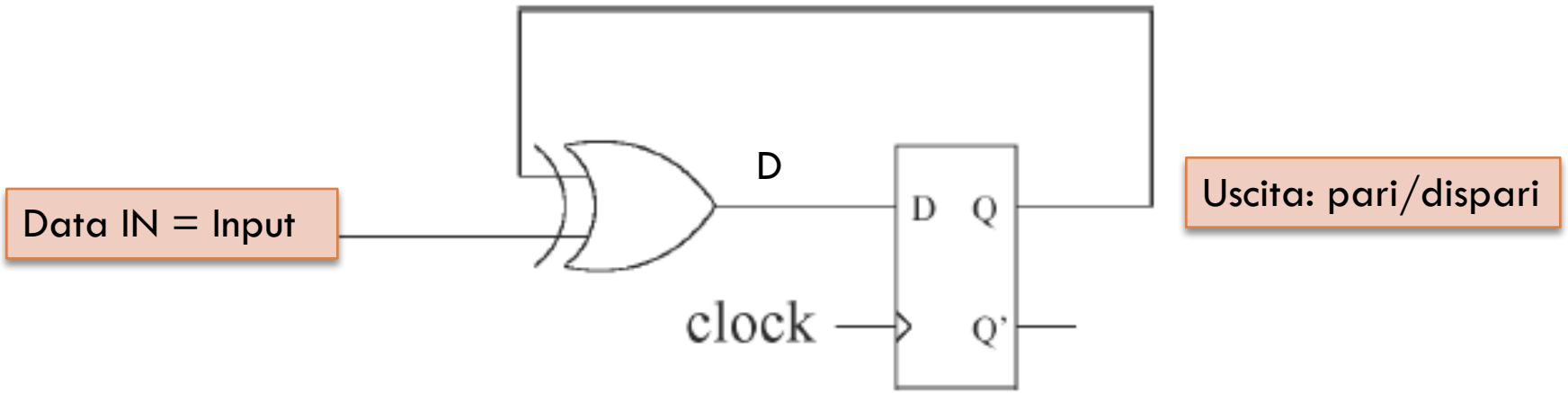
3. tabella della verita` dei valori delle uscite dello stato attuale
4. Scelta dei FF con cui voglio implementare la logica sequenziale -> valori degli ingressi per produrre i valori delle uscite che genereranno il prossimo stato
5.

Esempio di FSM: controllo di parita`

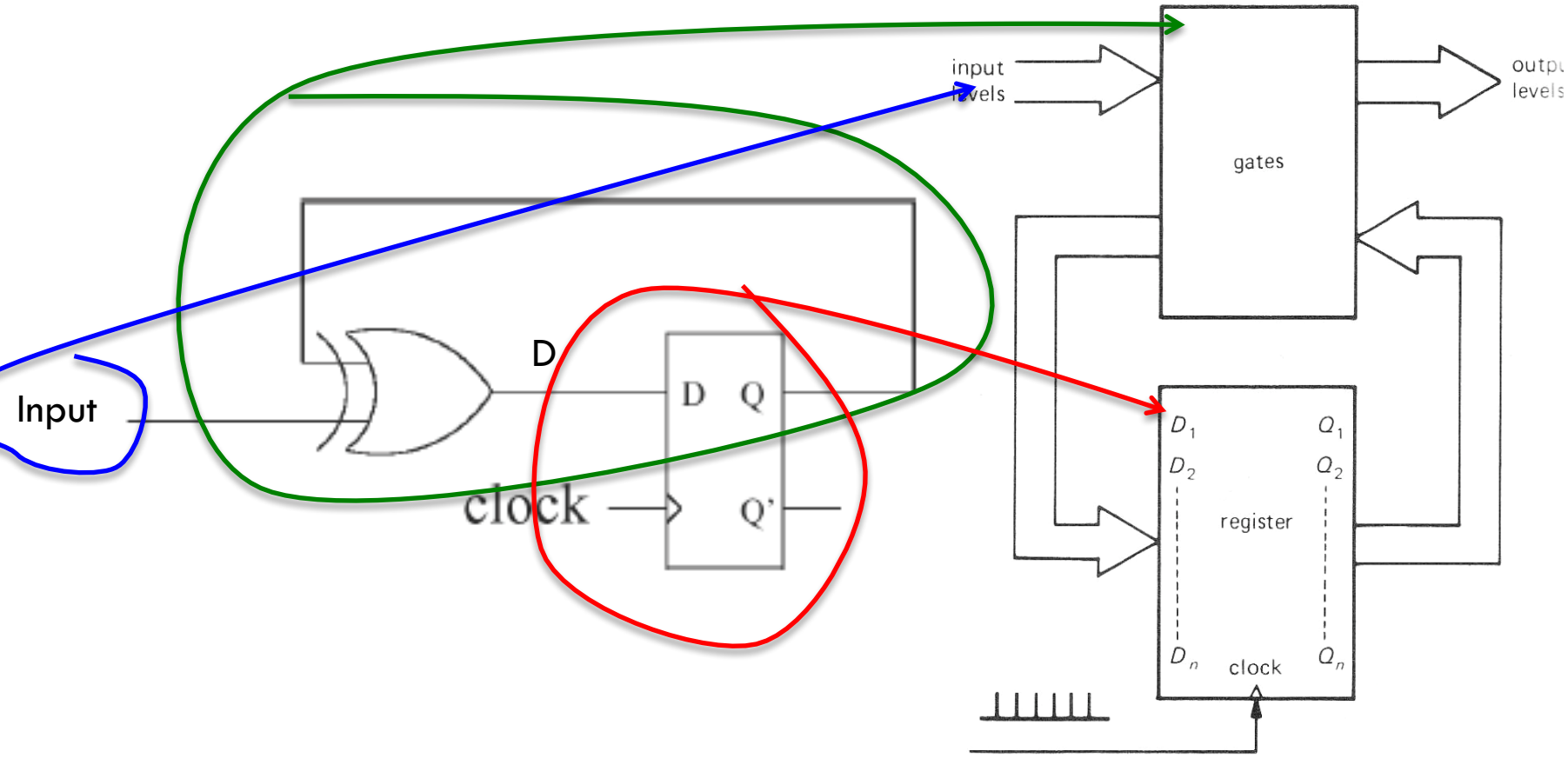
Q	Input	Q+	D
0	1	1	1
1	1	0	0
0	0	0	0
1	0	1	1

4. Trovare la rete combinatoria corretta per produrre gli ingressi dalle uscite available. Per fare questo si puo` anche utilizzare le mappe di Karnaugh

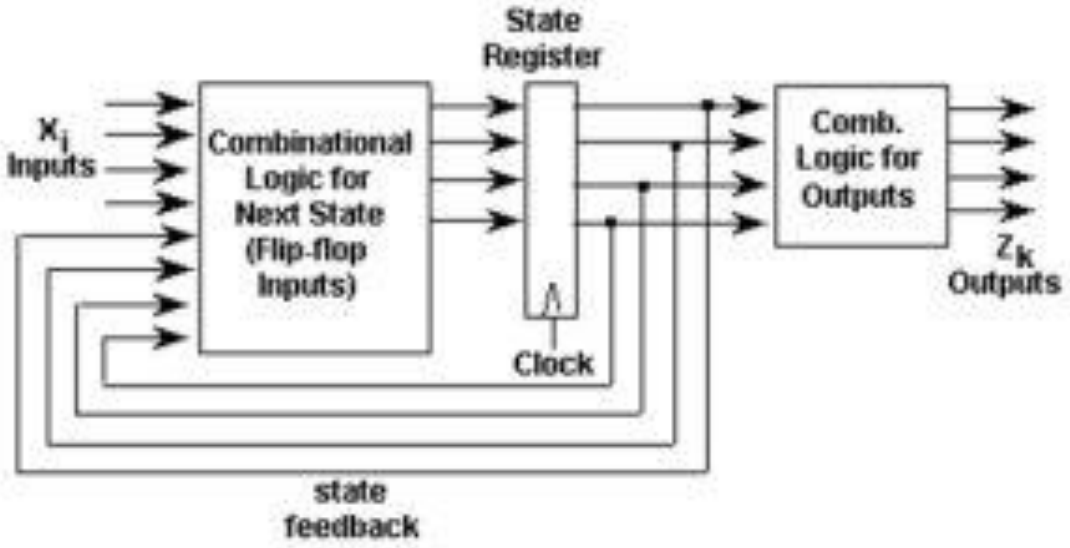
$$D = Q \oplus Input$$



Esempio di FSM: controllo di parita`



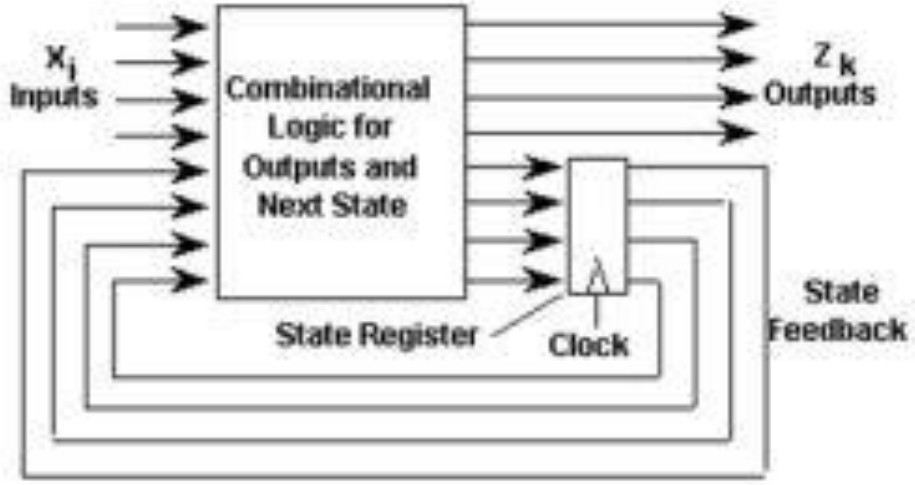
More and Mealy FSM



Moore Machine

Outputs are function solely of the current state

Outputs change synchronously with state changes



Mealy Machine

Outputs depend on state AND inputs

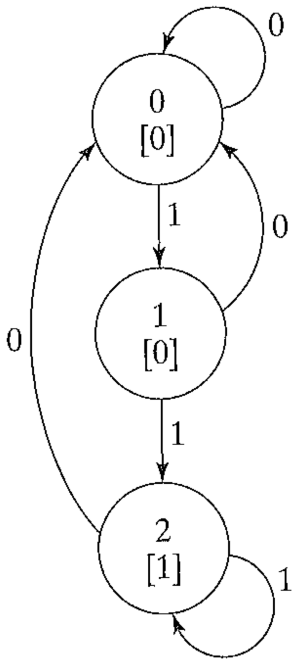
Input change causes an immediate output change

Asynchronous outputs

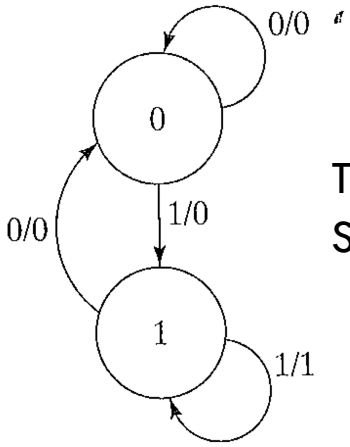
Esempio dei due tipi

- FSM per dare output 1 quando in ingresso arrivano almeno due "1" consecutivi.

Transizione: input
Stato: stato [output]



(a) Moore machine

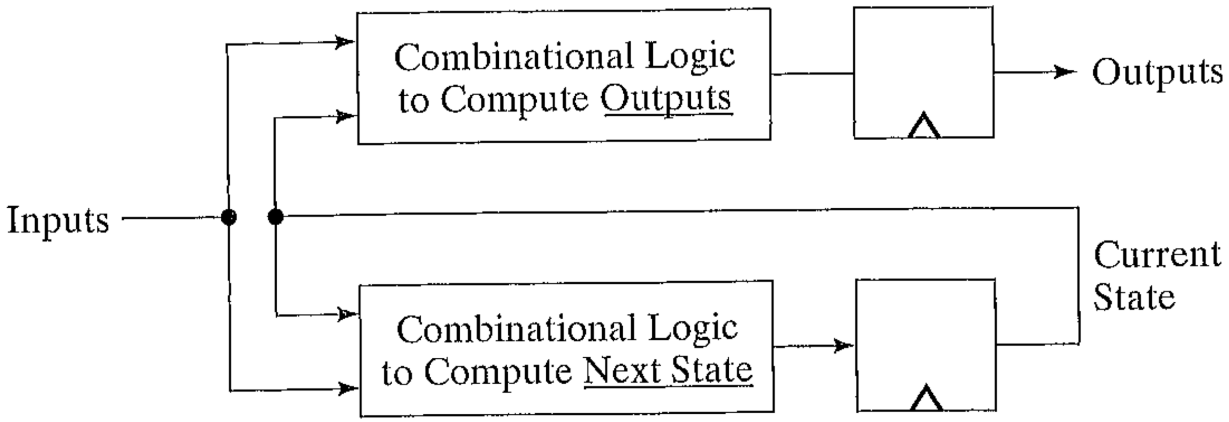


(b) Mealy machine

Transizione: input / output
Stato: stato

Sincrono/asincrono

- Se l'output cambia quando cambiano gli ingressi → Macchina asincrona
- Per sincronizzare la macchina di Mealy:
  - Registro sincrono sulle uscite



Progetto FSM

- Capire il problema
  - Tradurre dal linguaggio corrente in strutture più formali
- Costruire una rappresentazione astratta della FSM
  - Definire gli stati e le transizioni
- Minimizzare gli stati
  - Spesso si possono ridurre gli stati analizzando il problema. Ad es. le macchine di Mealy hanno spesso meno stati
- Definire la codifica degli stati nel registro
  - Può non essere ovvio
- Implementare la FSM
  - Logica combinatoria che definisce la matrice di transizione
- FSM si possono descrivere in HDL → sintetizzate

In generale: tabella di verità della FSM

State n	Input n	→	State n+1	Output
S0	I0		S1	O0
S0	I1		S0	O1
S1	I0		S0	O2
S1	I1		S1	O3

Spazio SxI = Tutti gli stati x tutti gli ingressi

Funzione di transizione: SxI → S

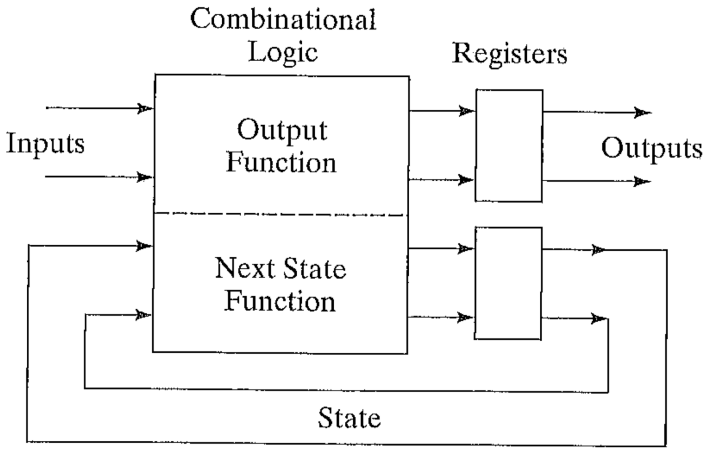
Uscite: dipende dal tipo di FSM
Moore: S → O
Measly: SxI → O

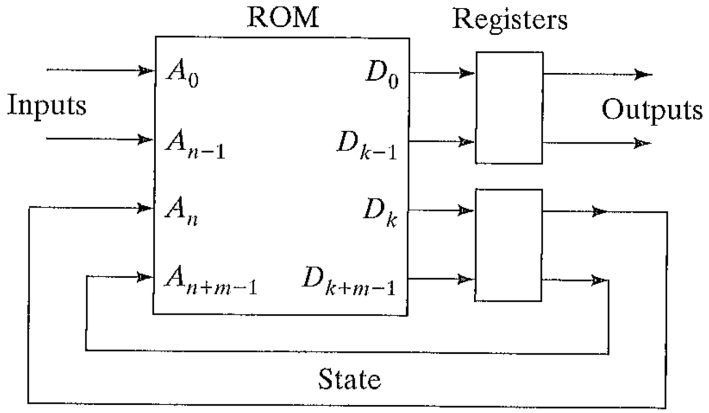
Implementazioni FSM

- Stato: registro
  - FF, Shift registers, counters
    - Dipende dalla struttura dell'ingresso (seriale, multiplo) e dal tipo di transizioni.
  - Importante il Load e reset.
- Matrice di transizione:
  - Porte logiche
  - ROM
  - PALs/PLAs
- FPGA che implementino sia lo stato che la transizione
  - Essenziale il software di programmazione

ROM per le FSM

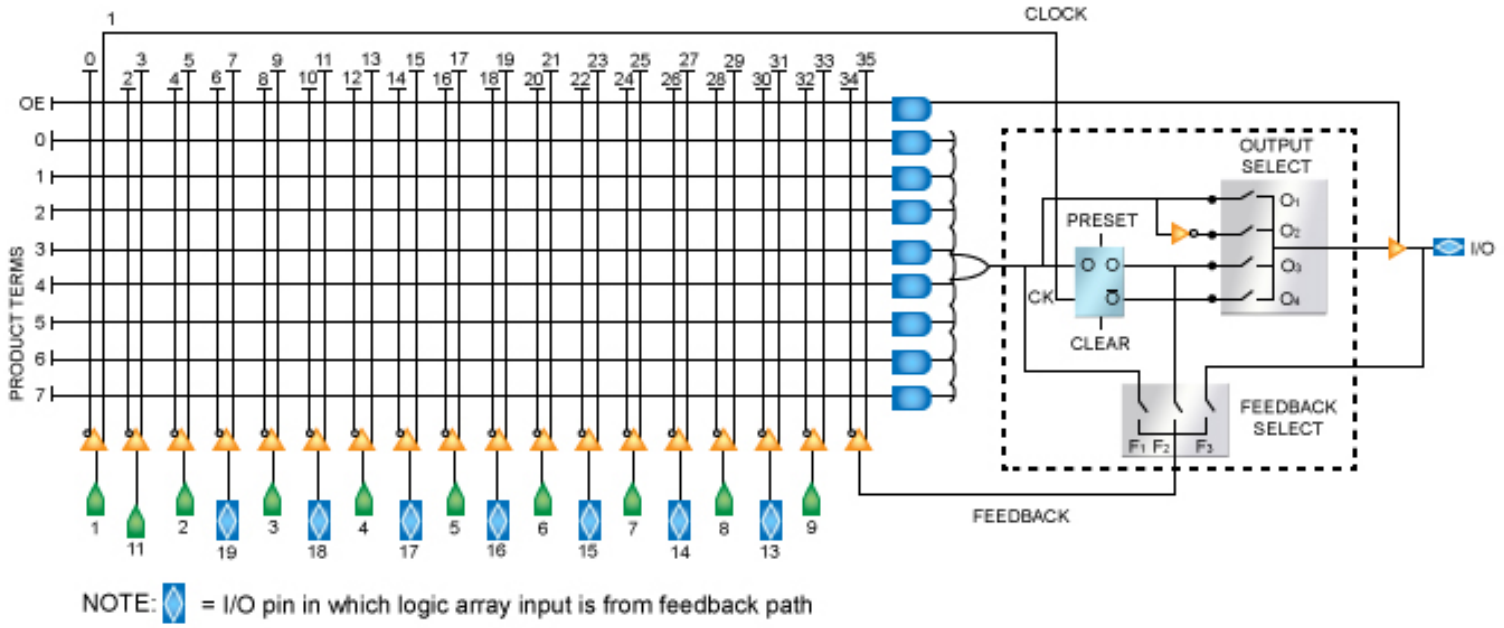
- Una ROM può essere usata facilmente per implementare la logica di transizione
  - Parte degli bit di indirizzo → input
  - Resto dei bit di indirizzo → State
  - Valore contenuto → Output e next state
- Esempio per Mealy FSM





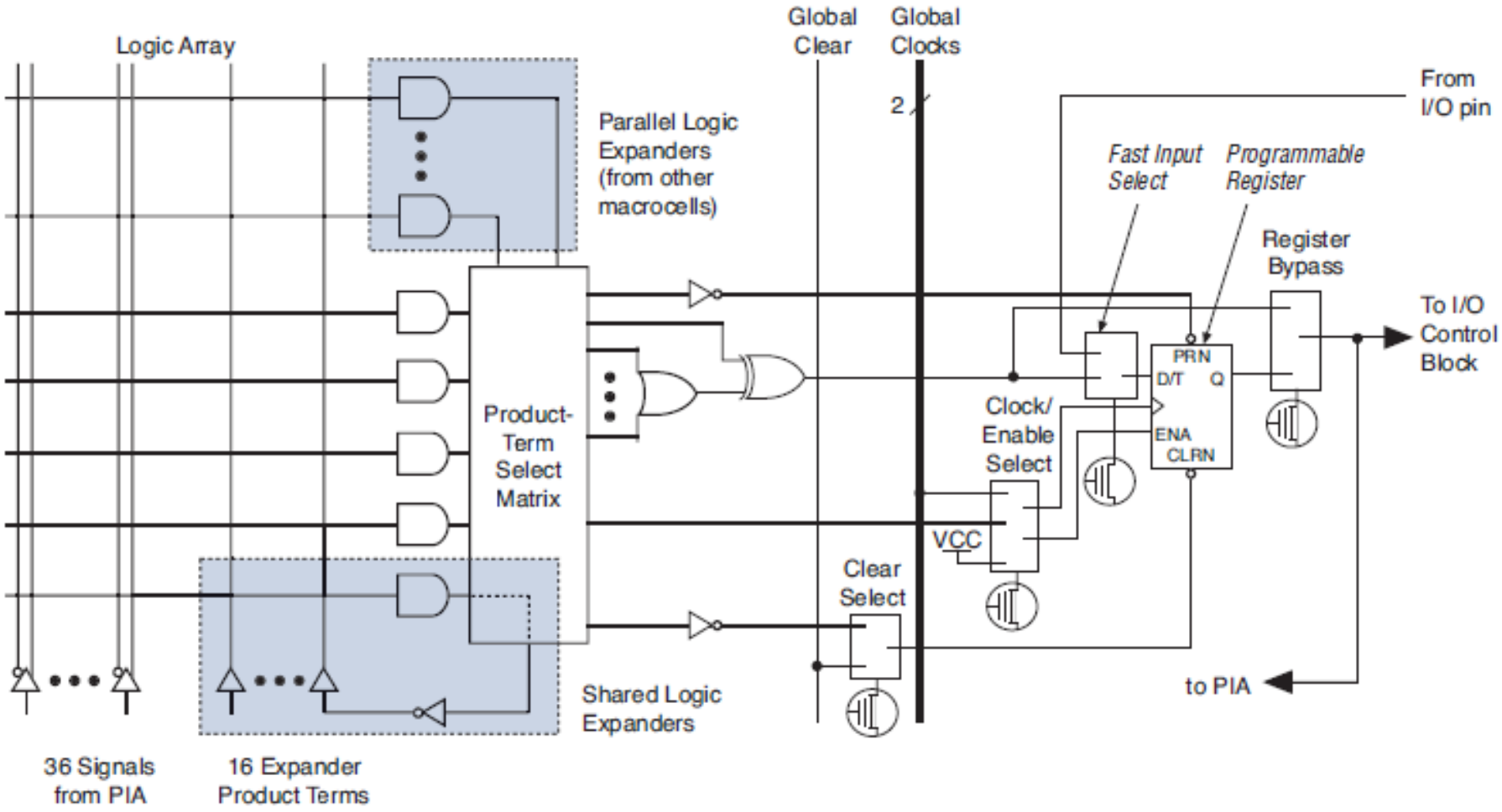
PALs / PLAs / PLDs

- Le PAL standard si possono usare per costruire la logica combinatoria di transizione
  - Comunque necessità di un registro esterno
- Registered PAL architecture (ad es. ALTERA)



This diagram shows one of the eight macrocells within the EP300/EP310.

Verso la complessità

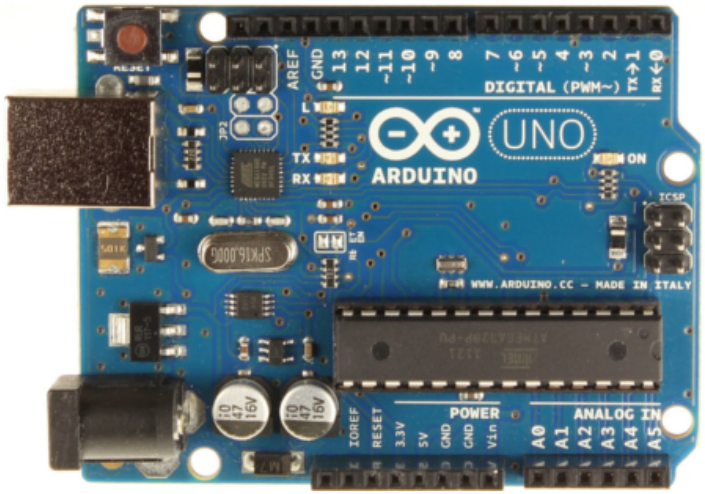


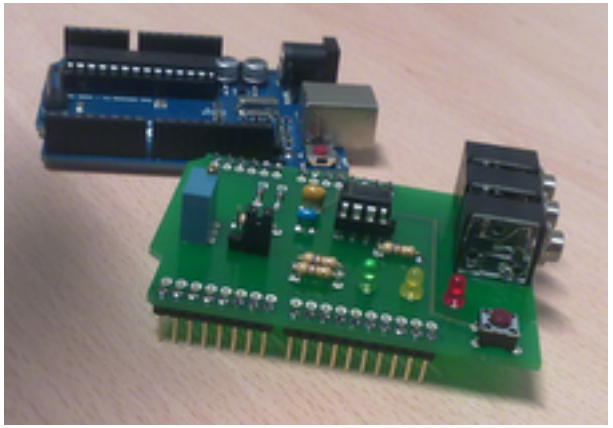
Elementi dei sistemi di controllo

- Input/output digitale e analogico (con signal conditioning)
- Parte logica veloce basata su FPGA
- Microprocessore per la programmazione e il controllo
- DSP (Digital signal processor) per l'elaborazione digitale di segnali convertiti

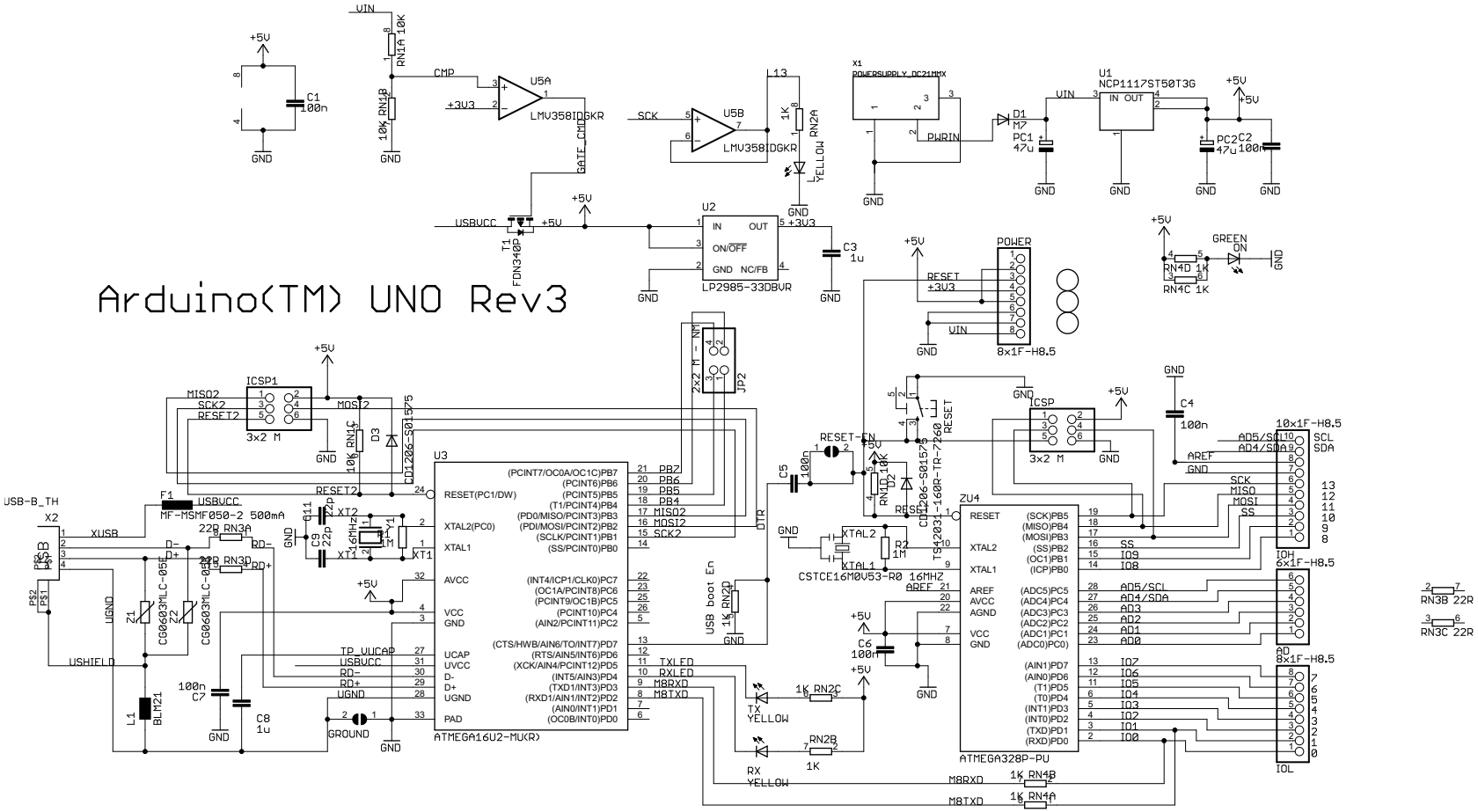
Schede con tutto dentro: arduino

- Sistema potente e aperto
  - http://www.arduino.cc/
- Introdotto nei lab didattici:
  - https://bitbucket.org/lbaldini/plasduino





Schema disponibile



Dove si trovano gli argomenti svolti oggi

- Introduzione all'elettronica Parte I – E.Falchini et al.
- Microelectronics – I.Millman
- Contemporary Logic Design – Katz, Borriello