www.delucagiovanni.com

Corsi, Forum, Docs – Robotics and A.I.

Introduzione alle Logiche Programmabili CPLD e FPGA

per la progettazione elettronica avanzata

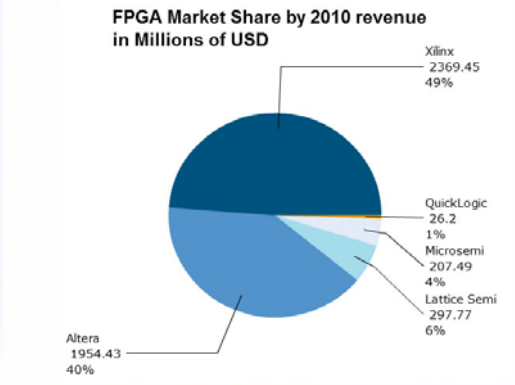
Giovanni De Luca

Gli acronimi

- *"Con i microcontrollori non si può fare tutto, in molti casi e in applicazioni particolari abbiamo necessariamente bisogno di realizzare hardware dedicato usando le poco conosciute **Logiche Programmabili**".*
- Alcune delle sigle e degli acronimi che si incontrano quando si ha a che fare con questi dispositivi sono queste:
- **CPLD** = **C**omplex **P**rogrammable **L**ogic **D**evices,
- **FPGA** = **F**ield **P**rogrammable **G**ate **A**rray,
- **JTAG** (**J**oint **T**est **A**ction **G**roup, IEEE1149) indica una particolare interfaccia e il relativo protocollo per la programmazione e il debug delle logiche programmabili.
- **Altera** è il nome di una delle aziende leader che producono i complessi chip delle logiche programmabili**.**

www.altera.com

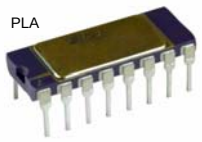
Il mercato delle FPGA



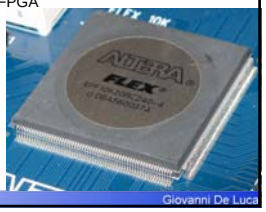
Programmable Logic Device (PLD)

- Dispositivo largamente utilizzato nei circuiti digitali per minimizzare il numero di integrati discreti (CD40XX e SN74XX) che realizzano logica combinatoria e sequenziale in una scheda elettronica.
- Non è configurato per svolgere una determinata funzione logica.
- Prima di poterlo utilizzare deve essere programmato.

PLA



FPGA

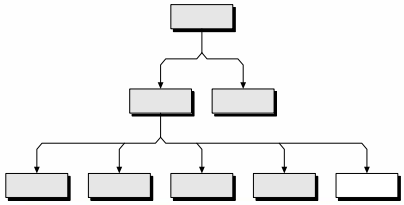


Programmable Logic Device

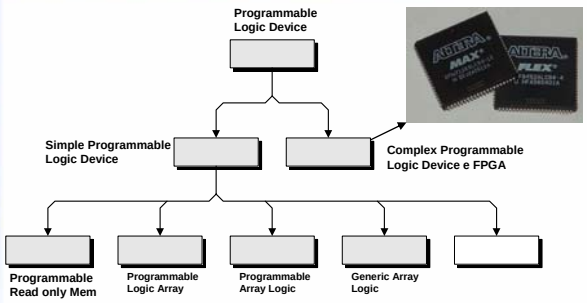
I PLD sono generalmente divisi in due categorie:

CPLD - Complex **P**rogrammable **L**ogic **D**evice ed i
SPLD – Simple **P**rogrammable **L**ogic **D**evice, a seconda che abbiano un numero di pin rispettivamente maggiore o minore di **48**.

Per i dispositivi più complessi (**CPLD**), tuttavia, si usa il termine **F**ield **P**rogrammable **G**ate **A**rray o **FPGA**.



Classificazione dei PLDs



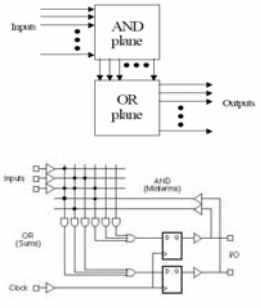
PLD - Famiglie

PROM
- Limitato numero di ingressi e uscite
- Non adatte per applicazioni veloci

PLA
- Maggior numero di ingressi e uscite
- Più veloci

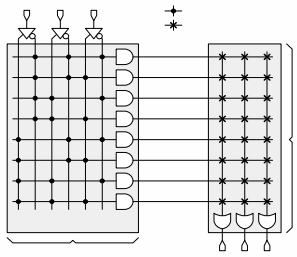
PAL
- Or-plane non configurabile
- Aggiunta di elementi di memoria sequenziali (flip-flop)

GAL
- Or-plane non configurabile come PAL
- Aggiunta di elementi di memoria e presenza di **Output Logic Macrocells**



Giovanni De Luca

SPLD con architettura PROM



La struttura tipica di un **SPLD** basato su una **ROM** programmabile è fatta da un piano **AND** predefinito, che corrisponde al decodificatore di riga di una **ROM**, e da un piano **OR** programmabile.

Giovanni De Luca

Uso di PROM come PLD

PROM (ROM programmabili)
EPROM (PROM cancellabili con raggi ultravioletti)
EEPROM (PROM cancellabili elettricamente)

Vantaggi: Facili da programmare (basta un semplice programmatore)

Limitazioni:
1 sono solitamente più lenti dei corrispondenti circuiti logici dedicati ovvero specific purpose (**ASIC** Application Specific Integrated Circuit),

2 non sono sempre immuni da errori in caso di transizioni asincrone,

3 consumano più potenza,

4 poiché sono spesso utilizzati solo per una piccola parte della loro capacità teorica, comportano un'eccessiva occupazione di spazio.

5 non possono essere facilmente impiegati per implementare circuiti a logica sequenziale poiché non dispongono della funzione **flip-flop**.

Giovanni De Luca

c

Predefined link
Programmable link

Address 0
Address 1
Address 2
Address 3
Address 4
Address 5

& !a & !b & !c
& !a & !b & c
& !a & b & !c
& !a & b & c
& a & !b & !c
& a & !b & c

mmable OR array

PLA (Programmable Logic Array)

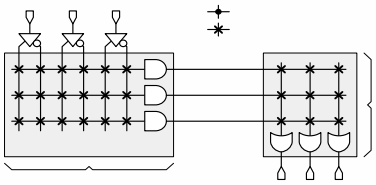
- E' il più semplice dispositivo logico programmabile
- Prodotto a livello industriale verso la fine degli anni '70
- Funzione logica descritta tramite ***somme di prodotti logici***.
- Composto da una matrice d'ingresso di porte logiche **AND** programmabili, collegate con una serie di **OR** programmabili.

L'architettura di un **PLA** comprende linee di retroazione dall'uscita all'array di porte **AND**, che possono essere utilizzate come ingressi aggiuntivi.
I primi dispositivi **PLA** ad essere stati prodotti implementano circuiti puramente combinatori, successivamente si sono sviluppati **PLA** sequenziali, che hanno la medesima architettura dei precedenti ma sono dotati di flip-flop per la sincronizzazione dei segnali.

SPLD con architettura PLA

Un **PLA** (*programmable logic array*) è simile a una PROM ma ha sia il piano AND che il piano OR programmabili.

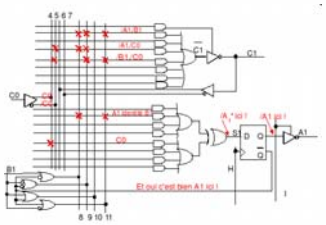
Lo svantaggio principale è una minore velocità perché il segnale in una **PLA** deve attraversare due livelli di collegamenti programmabili (che sono più lenti di un collegamento fisso).



Programmable AND array

PAL

I **PAL** sono un'evoluzione dei **PLA**, e differiscono principalmente per l'impossibilità di programmare la serie di porte **OR**.
Questi dispositivi presentano a seconda del modello, diverse soluzioni architetturali avanzate in più rispetto al suo predecessore.



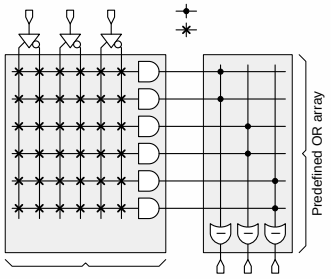
a b c

Predefined link

SPLD con architettura PAL

L'architettura **PAL** (*Programmable Array Logic*) è nata come tentativo di risolvere il problema della bassa velocità delle **PLA**, mantenendo la programmabilità del piano **AND**.

Il piano **OR** predefinito limita il numero di implicanti che contribuiscono ad ogni uscita (nell'esempio a fianco ci sono due implicanti per uscita).

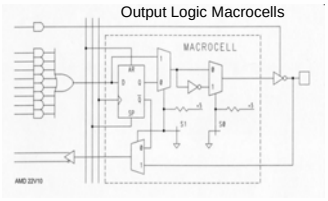


a b c

~~Predefined link~~
Programmable link

GAL (Generic Array Logic)

Il **G**eneric **A**rray **L**ogic, solitamente abbreviato in **GAL**, è un'evoluzione del **PAL** (Programmable Array Logic) ed è sviluppata dalla **Lattice Semiconductors**. Queste logiche hanno le stesse caratteristiche delle **PAL**, con l'aggiunta di dispositivi di output programmabili detti **OLMC** (Output Logic Macrocells)



&
&
&
&
&
&

a !a b !b c !c

Programmable AND array

w x y

CPLD (Complex Programmable Logic Array)

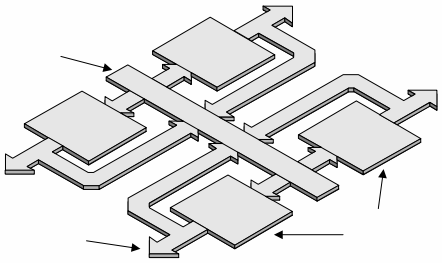
Mentre i **GAL** sono disponibili soltanto in piccole taglie, contenenti l'equivalente di alcune centinaia di porte logiche, i **CPLD** consentono di realizzare circuiti logici più complessi. Questi dispositivi contengono l'equivalente di molte **PAL** collegate fra di loro mediante interconnessioni programmabili ed incapsulate in un unico circuito integrato.

I **CPLD** possono equivalere a migliaia, a volte addirittura centinaia di migliaia di porte logiche.

Alcuni tipi di **CPLD** si programmano usando il «PAL programmer», ma questo metodo diventa poco pratico quando si devono collegare componenti con centinaia di pin. Un metodo molto più efficiente consiste nel saldare i dispositivi su un circuito stampato e quindi inviare loro, mediante un PC, un flusso di dati che, opportunamente decodificati dai circuiti interni dei **CPLD**, conferiscono agli stessi la configurazione necessaria a realizzare le funzioni logiche desiderate.

CPLD - PLD complessi

Concetto base: matrice di blocchi **SPLD** collegati da una rete di interconnessioni programmabili.

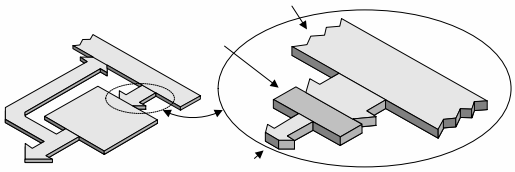


CPLD - le interconnessioni programmabili

I primi **CPLD** (*Mega-PAL* di *Monolithic Memories Inc.*) risultarono un fallimento a causa dell'eccessivo consumo di potenza e della bassa velocità, entrambi dovuti alla scelta di garantire il 100% di connettività

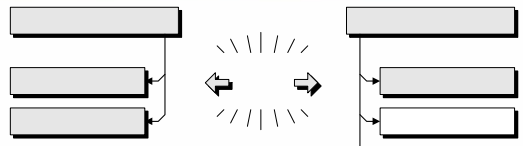
(ogni uscita di un blocco poteva essere connessa a qualunque ingresso di un altro blocco).

Altera propose una architettura con connettività incompleta, realizzata in tecnologia CMOS con celle di memoria EPROM.

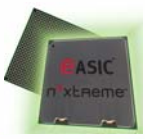


SPLD-like blocks

PLD e ASIC (Application Specific Integrated Circuit)



Subito dopo l'avvento dei primi **CPLD**, è emersa chiaramente una lacuna tra la categoria dei circuiti integrati programmabili, versatili e veloci da progettare e realizzare, ma limitati a funzioni relativamente semplici, e la categoria dei circuiti integrati per applicazioni specifiche (**ASIC**), che permettono di realizzare sistemi di grande complessità, ma richiedono una fase di progetto e realizzazione lunga e costosa, e non sono riconfigurabili.



FPGA (Field Programmable Logic Array)

82s100 – la prima **FPGA** prototipo venne prodotto dalla **Signetics** verso la fine degli anno 70 mentre nasceva la tecnologia gate-array.

L'**FPGA** utilizza una matrice di porte logiche molto simile a quella di un normale gate array, ma la programmazione è fatta dall'utilizzatore anziché in fabbrica.

La definizione di "**field programmable**" (cioè "programmabile sul campo") può sembrare poco chiara, ma il termine "**field**" significa semplicemente sul campo "**fuori dalla fabbrica**", quindi nel luogo dove viene utilizzato dall'utente finale.
Le **FPGA**, come i **CPLD**, di norma, si programmano dopo averli saldati al circuito stampato. La loro configurazione è volatile e deve essere ricaricata ad ogni riaccensione ed ogni volta che si richiede una diversa programmazione.

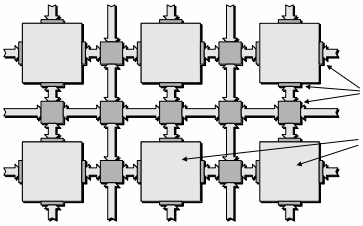
La prima FPGA

La famiglia più evoluta

Giovanni De Luca

Field Programmable Gate Array (FPGA)

- Componenti proposti a liv. industriale per la prima volta da **Xilinx** del 1984
- Realizzati in tecnologia **CMOS** con memoria di configurazione **SRAM**
- Architettura simile ai primi **CPLD** ma con un numero più elevato di blocchi logici programmabili più semplici, immersi in una griglia di interconnessioni con matrici di commutazione agli incroci



Giovanni De Luca

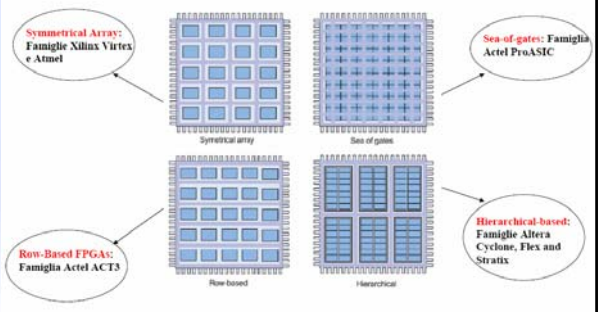
Cosa sono le FPGA ?

Una **Field Programmable Gate Array** si può semplicisticamente descrivere come un circuito riprogrammabile composto da array di flip-flop (**registri**) e di logica (**Look Up Table**).
Il programmatore decide come effettuare le connessioni tra logica e Flip Flop (**FF**), tra FF e FF, tra i pin di ingresso e di uscita ed i contenuti delle Look Up Table (**LUT**).

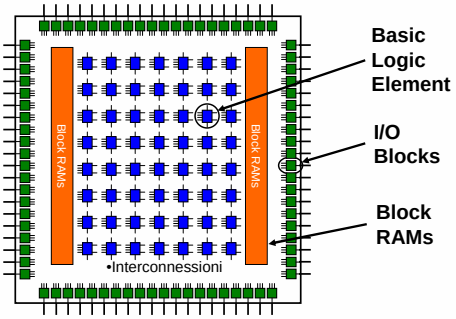
Oggi giorno questa definizione è molto semplicistica.
Come vedremo in seguito, infatti, le moderne **FPGA** integrano al loro interno diversi componenti hardware come moltiplicatori, blocchi DSP, serializzatori e deserializzatori, memoria, ADC e DAC ed interi microprocessori.

Giovanni De Luca

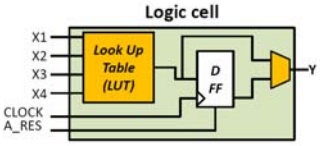
Diverse Architetture di FPGA



Come è composta una FPGA?



BLE (Basic Logic Element) e LUT



Modello generale di riferimento

I dettagli delle celle logiche in realtà variano in base al produttore ed al modello dell'FPGA.

Look-Up Table

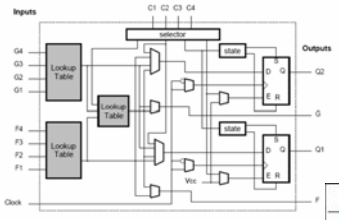
X1	X2	X3	X4	Y
0	0	0	1	?
0	0	1	0	?
...	...	...	...	?
1	1	1	1	?

Tipicamente ogni cella comprende:

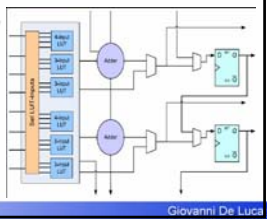
- **Una look-up table:** che consente di mappare una qualsiasi funzione combinatoria 4 ingressi 1 uscita
- **Un FF di tipo D** (con set e clr asincroni)
- **Un mux 2 -> 1:** per bypassare il FF in caso di celle puramente combinatorie

Elemento programmabile

Cosa sono le LUT

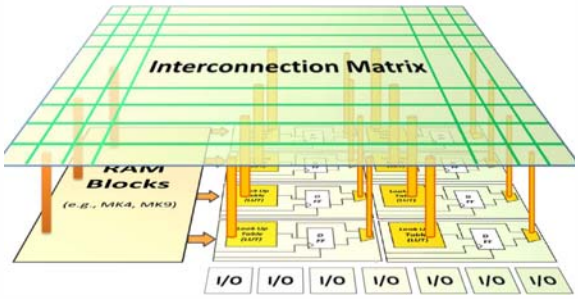


Una **LUT** consiste fisicamente in un insieme di celle **SRAM** per memorizzare i valori e un decoder che viene utilizzato per accedere alla corretta posizione **SRAM** e recuperare il risultato della funzione, che corrisponde alla combinazione di ingresso.

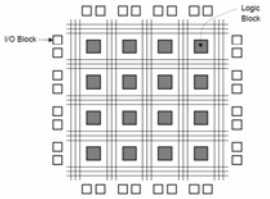


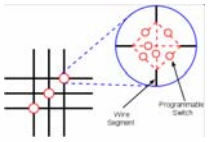
Le **LUT** sono delle tabelle di verità. Sulle vecchie FPGA avevano quattro ingressi ed una uscita. Su quelle moderne hanno 4 ingressi e 2 uscite. Nelle moderne FPGA arriviamo fino a 6 LUT e 8 FF per logic element.

Interconnessioni



Struttura delle FPGA e routing





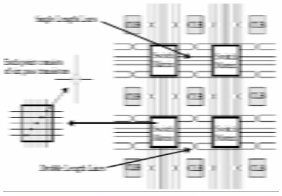
Possiamo considerare una **FPGA** come un insieme di risorse di routing, elementi di memoria (Flip Flop), elementi di logica (LUT) e risorse di I/O.

Gli elementi di logica e di memoria sono interconnessi tra di loro e con le risorse di I/O dalle risorse di routing. Degli switch programmabile (dalla memoria di configurazione) settano il comportamento dei **programmable switch**, creando il circuito vero e proprio in base alle decisione del programmatore.

Risorse di routing (FastTrack)

Le risorse di routing sono un elemento fondamentale di una **FPGA**.
Queste collegano i vari **logic element** tra di loro e con le risorse di I/O.
Si potrebbero immaginare queste risorse come dei collegamenti a matrici che corrono per tutta la **FPGA**, con switch che ne condizionano il percorso.
Alcuni di questi segnali sono privilegiati, in particolare le risorse relative alla gestione del clock.
I clock infatti sono un segnali particolari, che in generale devono arrivare in tutta la **FPGA** con il minor skew (massima differenza nel tempo di arrivo ai vari registri) e jitter possibile.

Le interconnessioni con ***FastTrack*** (typ. **Altera**):
- hanno minore complessità nella struttura del routing,
- le prestazioni sono predicibili con maggiore accuratezza.

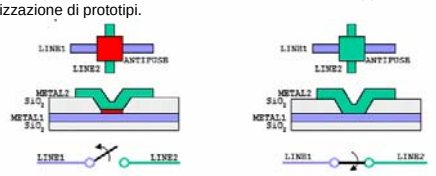


FPGA e CPLD: confronto

- **Complessità:**
  - **CPLD** da 1 Kgate a 100 Kgate
  - **FPGA** dal 10 Kgate a 10 Mgate
- **Granularità:**
  - **CPLD** da 1 a 100 blocchi logici programmabili di tipo PLA di dimensioni relativamente grandi
  - **FPGA** da 100 a 1Milione di blocchi di dimensioni ridotte (2 ÷ 6 ingressi, 1 o 2 registri), con una griglia di interconnessioni molto complessa.
- **Ritardi di propagazione:**
  - **CPLD,** i tempi di propagazione sono dominati dai blocchi logici e il contributo delle interconnessioni può essere calcolato a priori abbastanza accuratamente;
  - **FPGA,** il contributo delle interconnessioni domina i tempi di propagazione, e dipende fortemente dal piazzamento dei blocchi logici, per cui il ritardo non può essere stimato accuratamente a priori
- **Meccanismo di programmazione:**
  - **CPLD** è E^2PROM/FLASH
  - **FPGA** è SRAM o antifusibili

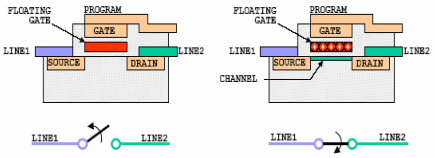
FPGA basate su Fuse ed Antifuse

FPGA fuse based :
L'informazione circuitale è contenuta utilizzando dei fusibile che rendono queste FPGA program-once.
L' **Antifusibile** al silicio è un elemento microelettronico che si comporta da isolante ma che successivamente può essere reso permanentemente ed irreversibilmente conduttore con appositi segnali elettrici. Si basa sulla perforazione di uno strato molto sottile (~9 nanometri) di nitruro di silicio, un dielettrico isolante, mediante un breve impulso (circa 1 millisecondo) dell'ampiezza di circa 16 volt.
È usato in memorie **PROM** e in dispositivi logici programmabili per creare i collegamenti elettrici tra i vari elementi logici.
Essendo componenti **OTP** non sono usati in ambienti di sviluppo e nella realizzazione di prototipi.



FPGA basate su Flash

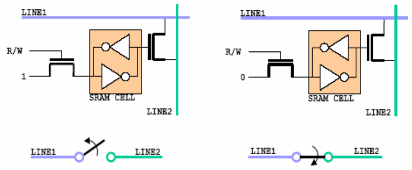
L'informazione è contenuta su una memoria **Flash** non volatile.
La memoria Flash può essere utilizzata direttamente per pilotare il circuito (es: **FPGA** Lattice) ed in questo caso il nostro circuito diventa disponibile immediatamente dopo il power on.
In altri casi un (die) di memoria flash è stato semplicemente integrato nel package della **FPGA** che dovrà comunque essere programmata, per cui non è immediatamente disponibile dopo il power-on.



- La programmazione consiste nel depositare carica sul floating gate del transistor in modo da mantenerlo in conduzione

FPGA basate su SRAM

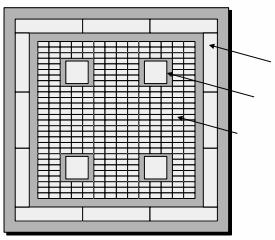
L'informazione è contenuta su una memoria volatile (RAM statica).
I singoli bit di questa memoria contengono la programmazione della nostra **FPGA**. Dato che la memoria **SRAM** è volatile ad ogni riaccensione è necessario riprogrammare questa memoria **SRAM**, tipicamente attraverso una memoria **FLASH** direttamente collegata alla **FPGA** esterna. (ma non è l'unico modo).
Tipicamente all'accensione, la **FPGA** provvede autonomamente (attraverso della logica cablata al suo interno) a scaricarsi il bit-stream dalla memoria **FLASH**.



FPGA: moduli funzionali specifici

Le FPGA spesso includono moduli funzionali specifici come:

- sommatori
- moltiplicatori
- unità MAC
- memorie RAM
- core di microprocessori
- interfaccie di I/O veloci



FPGA moderne

LE **FPGA** moderne, oltre ad avere una struttura della cella più articolata e con più risorse, includo al loro interno dei cosiddetti **Hard-IP**. Tipicamente questi sono:

Moltiplicatori, MAC: permettono avere moltiplicatori, MAC ed operazioni aritmetiche senza consumare risorse di logica della FPGA.
Un moltiplicatore implementato come LUT e FF richiederebbe molto spazio e sarebbe poco efficiente.

Dispositivi per la gestione del clock: il clock è una risorsa privilegiata all'interno della FPGA e per cambiarne il periodo, generare clock a frequenza differente ecc…, sono necessari dei componenti appositi chiamati PLL (Phase Locked Loop).

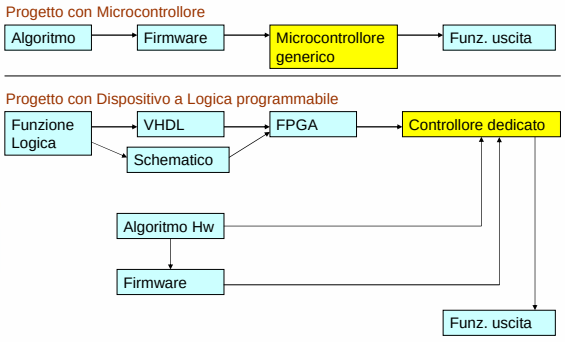
RAM: per immagazzinare dati senza avere un componente di memoria esterno alla nostra FPGA, i costruttori annegano dentro la FPGA più blocchi di memoria RAM che in totale possono arrivare a svariati Mbit.

Altera - Quartus II

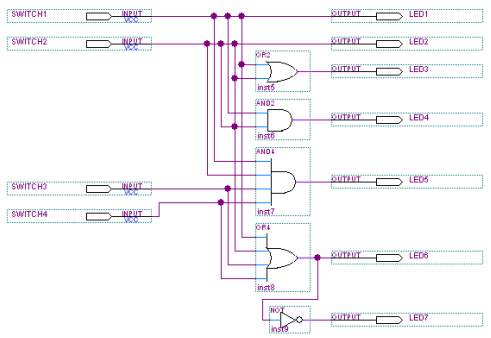
QUARTUS II è un tool utilizzabile per effettuare,
nell'ambito della progettazione di circuiti digitali le seguenti fasi:

- Descrizione di un circuito in Schematic o in VHDL entry
- Sintesi logica
- Simulazione digitale
- Place and Route
- Analisi delle prestazioni
- Programmare il dispositivo

Differenze flusso progettuale



Approccio “Schematic Entry”



Approccio descrittivo “VHDL”

VHDL e’ l’acronimo di **V**HSIC **H**ardware **D**escription **L**anguage

VHSIC e’ l’acronimo di **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit

- Nasce nel 1987 dal Dipartimento Difesa Americano
- E’ il linguaggio più usato oggi per la progettazione di sistemi elettronici digitali
- Il VHDL si presenta simile a un vero e proprio linguaggio di programmazione

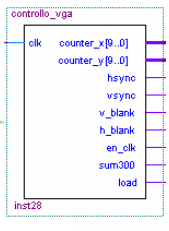
L’ aspetto più importante è la Concorrenzialità:

- Le diverse parti di un codice scritto in **VHDL** devono essere immaginate funzionanti contemporaneamente al contrario di un linguaggio software, dove le funzioni descritte dal codice sono generalmente eseguite dall'alto in basso, riga dopo riga in modo sequenziale.

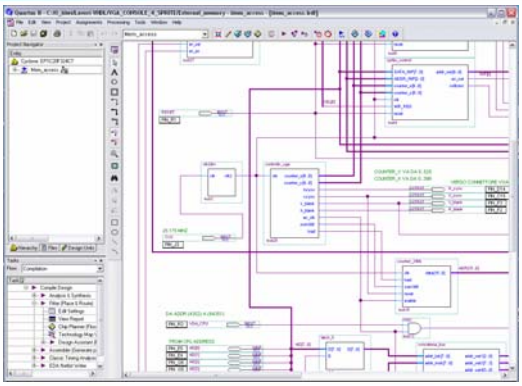
```
FLIPFLOP : Process (clock, reset)
-- Il processo viene eseguito se clock o reset cambiano valore
begin
if (reset='1') then
  Q <= '0';
elsif (clock'event and clock='1') then
  Q <= D;
end if;
end process;
```


VHDL to Symbol

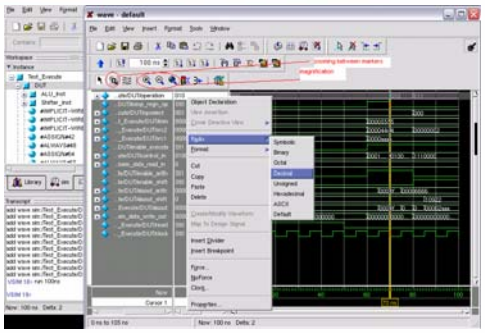
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity controllo_vga is
    Port ( counter_x:   out std_logic_vector(9 downto 0);  -- posizione assoluta in x 0..399
           counter_y:   out std_logic_vector(9 downto 0);  -- posizione assoluta in y 0..525
           clk :   in  STD_LOGIC;
           hsync :  out  STD_LOGIC;
           vsync :  out  STD_LOGIC;
           v_blank: out std_logic;
           h_blank: out std_logic;
           en_clk : out STD_LOGIC;
           sum300:  out std_logic;
           load:    out std_logic);

end controllo_vga;

architecture Behavioral of controllo_vga is
    constant max_c : integer := 799;  -- secondo contatore ausiliario
    constant max_x : integer := 399;  -- risoluzione orizzontale 399
    constant max_y : integer := 525;  -- risoluzione verticale 525

    signal c_counter: integer range 0 to max_c;
    signal h_counter: integer range 0 to max_x;    -- contatore orizzontale
    signal v_counter: integer range 0 to max_y;    -- contatore verticale
    signal fineriga: std_logic;
    signal enable_h: std_logic;
    signal enable_v: std_logic;
    signal xx:std_logic;
```



Schematic Entry con VHDL Symbols



Modelsim

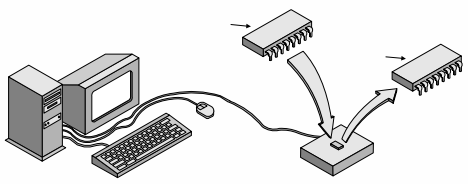


MODELSIM è un tool per la simulazione dei circuiti descritti in **VHDL**

FPGA/CPLD: programmazione del dispositivo

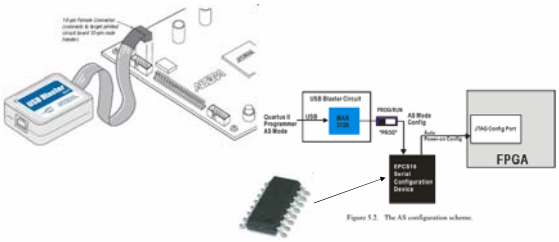
Il risultato della fase di progetto è un insieme di bit di configurazione (***configuration bit-stream***), che viene caricato nel componente attraverso un dispositivo di programmazione o semplicemente attraverso terminali dedicati sul componente stesso.

I bit di configurazione fissano lo stato (aperto/chiuso) degli interruttori che determinano la funzione svolta dai blocchi programmabili e la topologia delle interconnessioni.

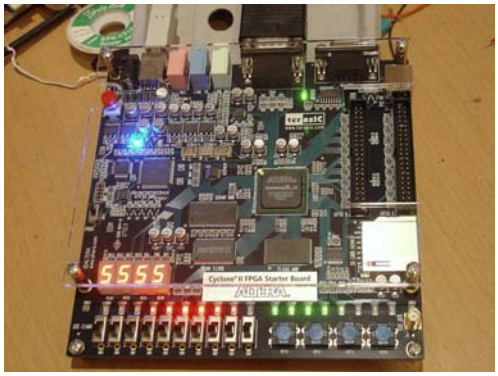


Configurazione e Bit-stream

La configurazione del circuito della FPGA deve essere immagazzinata su una memoria. Questa memoria contiene le informazioni su come sono configurati i vari switch interni per il routing globale e locale, il contenuto delle LUT, ecc … (Bit-stream)

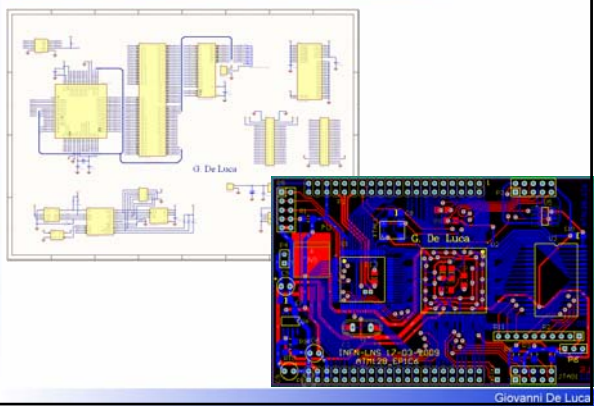


Demo Board - ALTERA

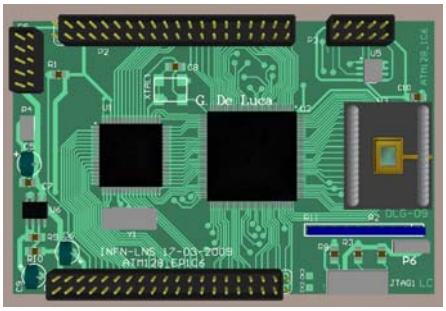


Altium Designer 10

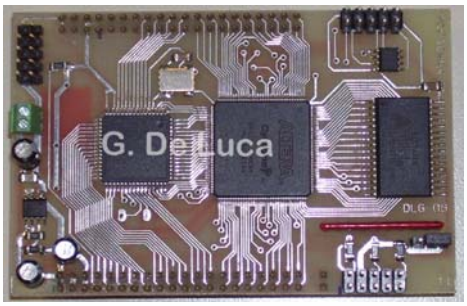
Dallo schema al PCB



Progetto 3D con FPGA



Scheda definitiva



Limiti attuali ?

- Massima Frequenza del Core = **1.5 Ghz**
- Minimo consumo stand-by = **25 uA**
- Tecnologia costruttiva = **22 nanoMetri**
- Data rate transceiver = **>10 GigaBits/sec**
- Max Numero Pin = **~1700 pin totali (BGA)**
- Logic cell = **~un milione sulla Virtex 6 (Xilinx)**
- Logic element = **~un milione sulla Stratix V** (Altera)

Equivalenti a:

~ 100 uControllori Atmega / PIC

~ 5 uProcessori Intel (I7)

FPGA - dove si trova e a cosa serve

- Automotive
- Mobile
- Intelligent Robotics
- Personal Computer
- Video Games Console
- Elettrodomestici
- Calcoli paralleli veloci
- ANN - Reti neurali artificiali
- Elettronica spaziale, satellitare e nucleare

Costi ?

- **FPGA / CPLD** – da pochi euro in su
- **Sistema di sviluppo educational** – gratuito
- **Programmatore USB/Jtag** – da 50 euro
- **Scheda di sviluppo** x EDU – da 150 euro
- **Schede prototipo** – da 25 euro

www.delucagiovanni.com

Introduzione alle
Logiche Programmabili
CPLD e FPGA

FINE

Giovanni De Luca