



Field Programmable Gate Array (FPGA)

Sistemi di Elaborazione Accelerata, Modulo 1

A.A. 2025/2026

Stefano Mattoccia

Università di Bologna

Cenni storici

- '70s: gates, SSI, MSI, LSI
- Successivamente: dispositivi programmabili (a livello di funzioni logiche, connessioni) anche sul campo come EPROM, PAL, PLA, PLD (studiati nel corso di Reti Logiche)
- Anni 80/90: introduzione di dispositivi logici ad elevata scala di integrazione di FPGA (e CPLD), programmabili sul campo mediante linguaggi HDL

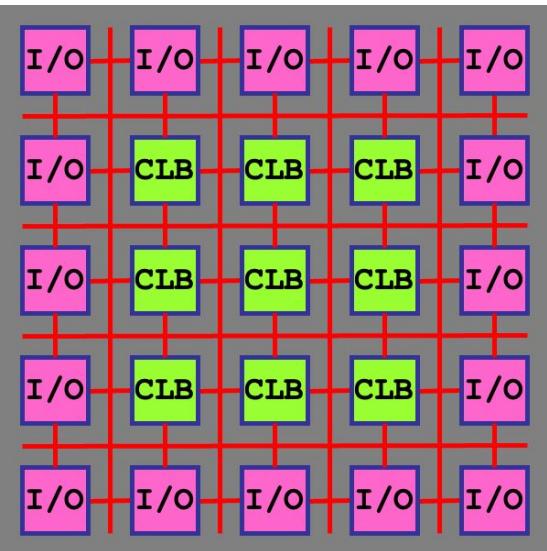


Caratteristiche dei dispositivi FPGA

- **FPGA**: composti da blocchi logici configurabili (Configurable Logic Blocks – CLB) che possono essere interconnessi tra loro in modo trasparente all'utente utilizzando linguaggi di programmazione di alto livello (e.g. HDL o HLS)
- Programmabili (e riprogrammabili) sul campo
- Relativamente poco costosi, rapido time to market, supportati da linguaggi di alto livello (C/C++) o Hardware Description Language (VHDL o Verilog)
- Ideali per lo sviluppo rapido di prototipi (e.g. usati per lo sviluppo di microprocessori)
- Ideali per dispositivi a basso consumo
- Consentono un elevato livello di astrazione -> è possibile implementare in hardware non solo reti logiche tradizionale ma anche algoritmi
- E' possibile integrare cores (sia soft che hard)
- FPGA vs ASIC (\$\$) – in entrambi i casi possono essere utilizzati linguaggi HDL

Moduli principali dei dispositivi FPGA

Una FPGA consiste in un insieme di Configurable Logic Blocks (CLB) che possono essere connessi tra loro. La funzione dei singoli CLB e delle connessioni viene impostata dal progettista mediante programmazione (“sul campo”). Tale programmazione può essere ripetuta più volte (teoricamente “infinite”).



- Numero di CLB molto elevato (migliaia)
- Generalmente è anche disponibile della RAM interna - Block RAM (centinaia di KByte)
- Alcuni blocchi sono dedicati all'I/O
- Disponibili (decine) di sommatori, moltiplicatori

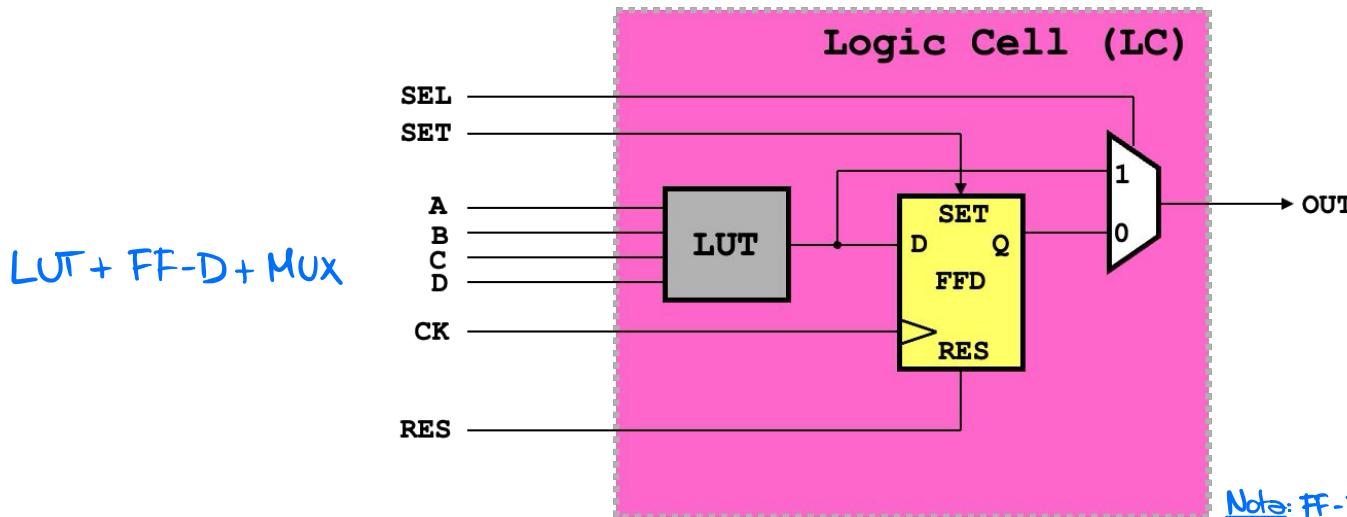
Tecnologie che caratterizzano i dispositivi FPGA

- Esistono diversi produttori di FPGA e differenti tecnologie
- FPGA di produttori diversi si differenziano principalmente per due aspetti principali:
 - **Tecnologia utilizzata per le connessioni**
 - Fusibili
 - Memorie flash
 - Memorie SRAM
 - **Struttura dei CLB**

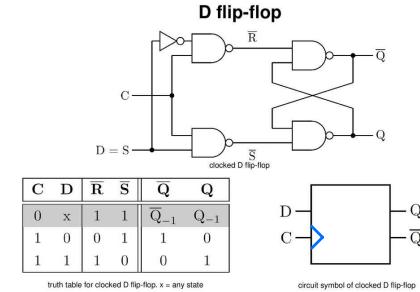
Sebbene un'analisi dettagliata dei blocchi logici configurabili (CLB) di una tipica FPGA esuli dagli obiettivi di questo corso, è molto interessante capire come tali blocchi sono organizzati

Logic Cell come elemento base dei CLB

- Lo schema seguente mostra, in forma semplificata, una rete denominata Logic Cell (LC) che potrebbe essere alla base di un ipotetico CLB

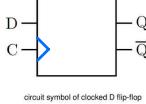


- Il blocco LUT (Look-up-table) non è altro che una rete combinatoria programmabile (vedi pg successiva)
- Il blocco LUT, può essere anche riprogrammato per agire come uno shift-register o un'una memoria (distributed RAM)



C	D	\bar{R}	\bar{S}	Q	\bar{Q}
0	x	1	1	\bar{Q}_{-1}	Q_{-1}
1	0	0	1	1	0
1	1	1	0	0	1

truth table for clocked D flip-flop: x = any state



November 2018

Electronics for physicists

Marc Weber - KIT

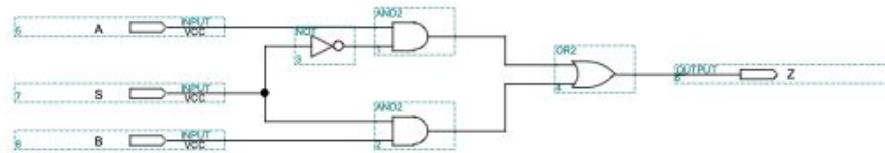
9

FF-D funziona come un latch
ma è clock-triggered
=> espone il valore in ingresso
al fronte di salita del clock

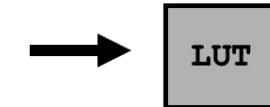
Note: FF-D deve rispettare setup time e hold time.
(cioè D stabile prima del fronte e dopo, per un t_s/t_h)

Funzioni combinatorie e Look-Up-Table (LUT)

- Una LUT può essere utilizzata per realizzare una qualsiasi rete combinatoria



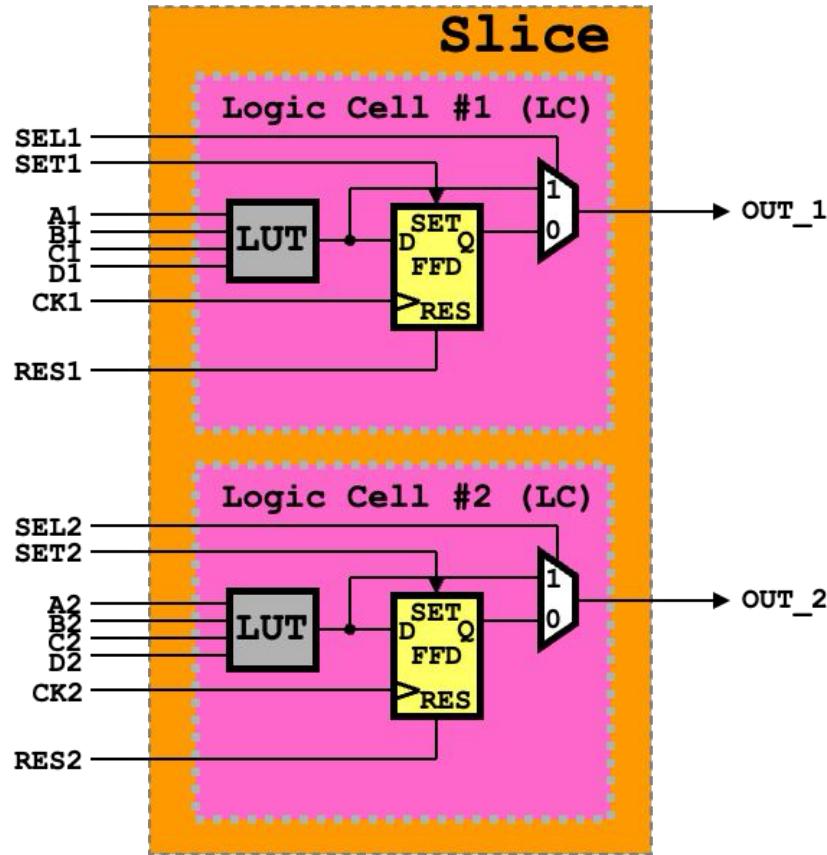
S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



- Oppure per essere configurata come un convenzionale elemento di memoria (distributed RAM)

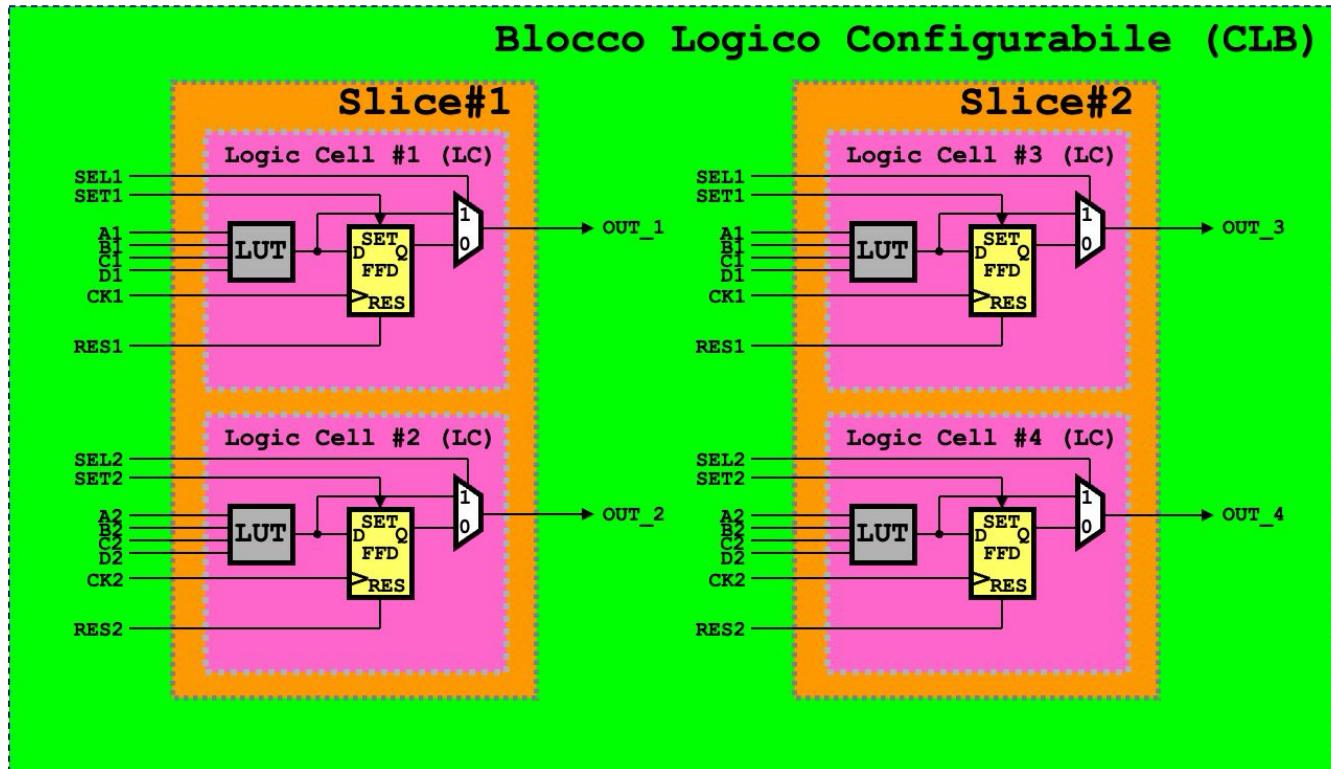
Slices come combinazione di Logic Cells

- Tipicamente più Logic Cell sono raggruppati per formare degli Slice



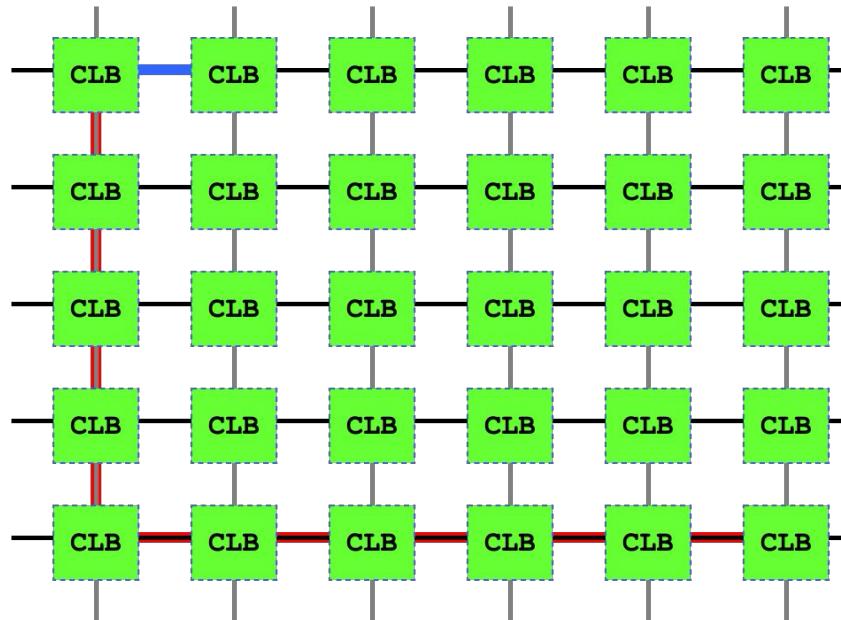
CLB come combinazione di Slice

- Infine, più Slice sono raggruppati per formare un CLB (Configurable Logic Block)



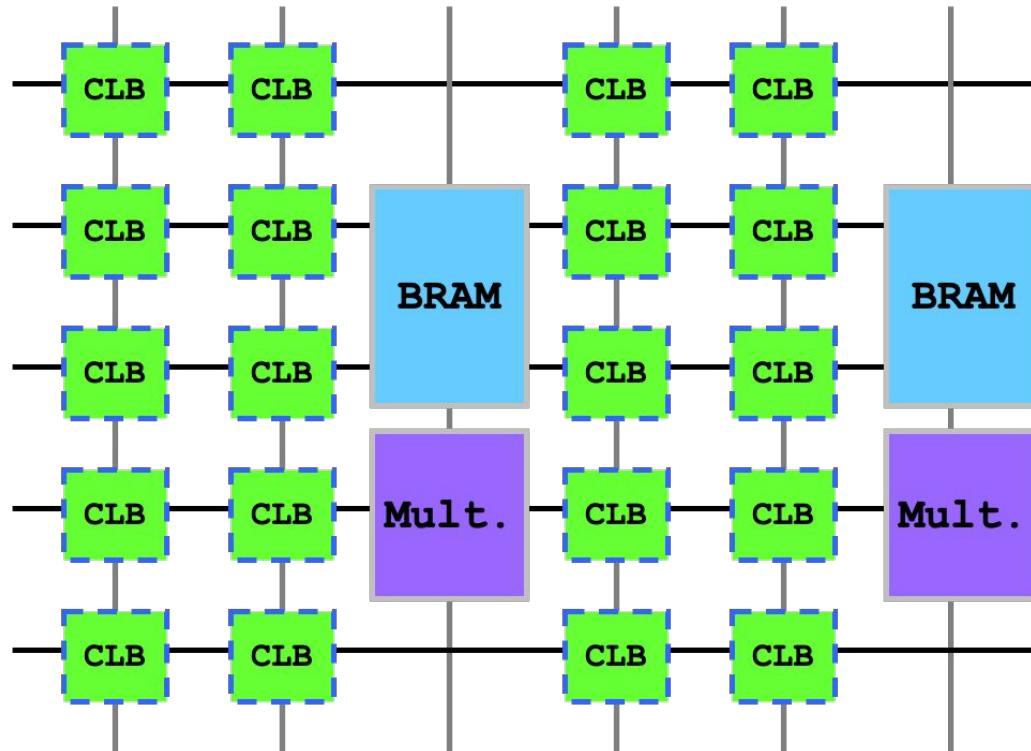
CLB e connessioni

- All'interno della FPGA esistono delle connessioni configurabili tra i vari CLB, per i clock, etc
- Le connessioni sono molte ma non infinite; inoltre, sorgono problematiche di routing/ritardi causati da differenti percorsi (e.g. rosso e blu)



CLB e moduli addizionali

- In realtà, tra i vari CLB sono presenti altre unità funzionali (Block RAM, moltiplicatori,...)



Esempio di FPGA reale

Spartan-6 FPGA Feature Summary

Table 1: Spartan-6 FPGA Feature Summary by Device

Device	Logic Cells ⁽¹⁾	Configurable Logic Blocks (CLBs)			DSP48A1 Slices ⁽³⁾	Block RAM Blocks		CMTs ⁽⁵⁾	Memory Controller Blocks (Max) ⁽⁶⁾	Endpoint Blocks for PCI Express	Maximum GTP Transceivers	Total I/O Banks	Max User I/O
		Slices ⁽²⁾	Flip-Flops	Max Distributed RAM (Kb)		18 Kb ⁽⁴⁾	Max (Kb)						
XC6SLX4	3,840	600	4,800	75	8	12	216	2	0	0	0	4	132
XC6SLX9	9,152	1,430	11,440	90	16	32	576	2	2	0	0	4	200
XC6SLX16	14,579	2,278	18,224	136	32	32	576	2	2	0	0	4	232
XC6SLX25	24,051	3,758	30,064	229	38	52	936	2	2	0	0	4	266
XC6SLX45	43,661	6,822	54,576	401	58	116	2,088	4	2	0	0	4	358
XC6SLX75	74,637	11,662	93,296	692	132	172	3,096	6	4	0	0	6	408
XC6SLX100	101,261	15,822	126,576	976	180	268	4,824	6	4	0	0	6	480
XC6SLX150	147,443	23,038	184,304	1,355	180	268	4,824	6	4	0	0	6	576
XC6SLX25T	24,051	3,758	30,064	229	38	52	936	2	2	1	2	4	250
XC6SLX45T	43,661	6,822	54,576	401	58	116	2,088	4	2	1	4	4	296
XC6SLX75T	74,637	11,662	93,296	692	132	172	3,096	6	4	1	8	6	348
XC6SLX100T	101,261	15,822	126,576	976	180	268	4,824	6	4	1	8	6	498
XC6SLX150T	147,443	23,038	184,304	1,355	180	268	4,824	6	4	1	8	6	540

Notes:

1. Spartan-6 FPGA logic cell ratings reflect the increased logic cell capability offered by the new 6-input LUT architecture.
2. Each Spartan-6 FPGA slice contains four LUTs and eight flip-flops.
3. Each DSP48A1 slice contains an 18 x 18 multiplier, an adder, and an accumulator.
4. Block RAMs are fundamentally 18 Kb in size. Each block can also be used as two independent 9 Kb blocks.
5. Each CMT contains two DCMs and one PLL.
6. Memory Controller Blocks are not supported in the -3N speed grade.

FPGA e prestazioni

- Le frequenze di funzionamento delle FPGA non sono molto elevate se confrontate a quelle dei processori (alcune centinaia di MHz, spesso più basse).
- Come è possibile essere competitivi con frequenze di funzionamento così ridotte?
- Configurando la FPGA per sfruttare al massimo il parallelismo
- Una nota (molto) positiva di questo approccio; riducendo la frequenza si riduce l'assorbimento di potenza ma con elevato parallelismo si ottengono eccellenti prestazioni

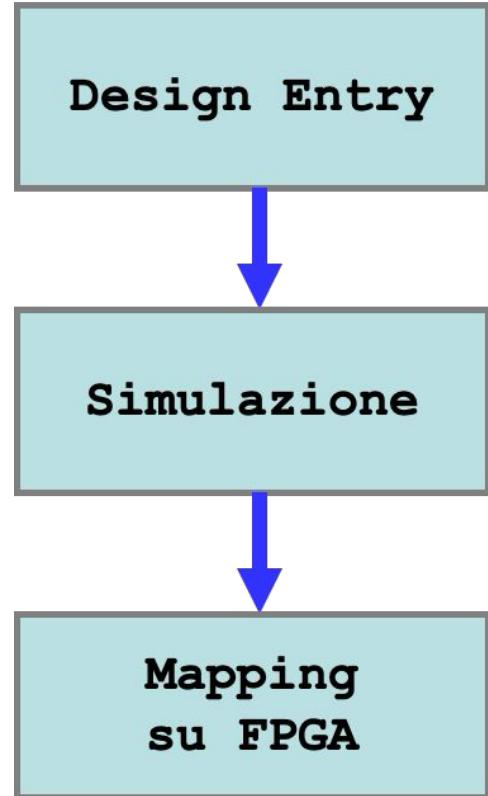
FPGA: linguaggi di programmazione/progettazione

- Come configurare una FPGA connettendo i vari CLB, etc al fine di realizzare un determinata logica?
- Per questa finalità (sintesi) si utilizzano i linguaggi HDL o tool di sintesi ad alto livello (HLS) e compilatori che generano i dati di configurazione (bitstream)
- Il bitstream è un sorta di software che configura la FPGA per svolgere un determinato compito/rete
- Il bitstream è generato a partire dal codice HDL (o dallo schematico, anche se oggi non più) o da tool di sintesi ad alto livello (quasi) senza l'intervento del progettista/programmatore (a meno che non desideri esplicitamente intervenire ma tipicamente questo non accade)

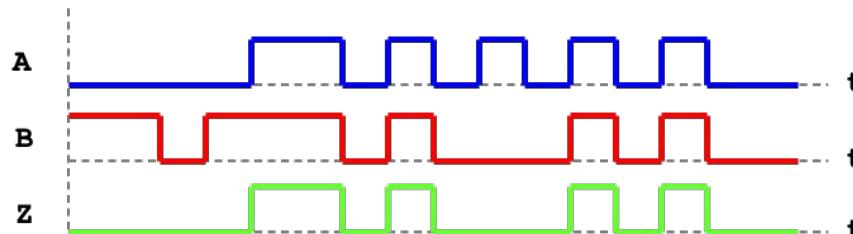
Progettazione con linguaggi HDL 1/2

- Linguaggi ad alto livello finalizzati alla simulazione di circuiti digitali ma sempre più utilizzate (anche) per la sintesi
- L'obiettivo è quello di ridurre i tempi e i costi di sviluppo, consentendo la gestione di progetti di grandi dimensioni, etc
- I linguaggio **HDL** (e.g. VHDL o Verilog) consentono di modellare il comportamento dell'hardware, ovvero: **intrinseco parallelismo e ritardi**
- Tipicamente utilizzati per programmare gli FPGA
- Utilizzati anche per realizzare ASIC, processori, etc
- Standard IEEE

Progettazione con linguaggi HDL 2/2



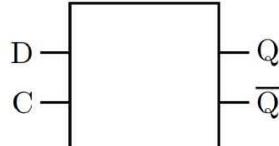
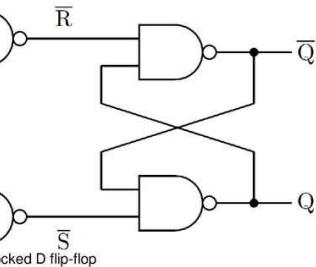
```
entity my_AND is  
Port (A : in BIT;  
      B : in BIT;  
      Z : out BIT);  
      . . .
```



Tutte e fasi sono eseguite al calcolatore

Progettazione con strumenti di High Level Synthesis (HLS)

flip-flop



circuit symbol of clocked D flip-flop

```
#include <math.h>

int function()
{
    int i;
    int r = 0;

    for (i=0;i<100;i++)
    {
        . . .
        . . .
    }
    return r;
}
```

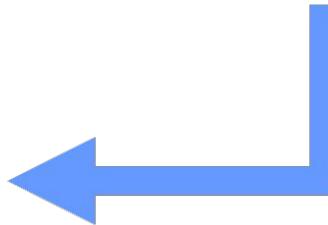
C++, etc



```
entity function is
Port(x : in type;
      Y : in type
      r : out type );
end function;

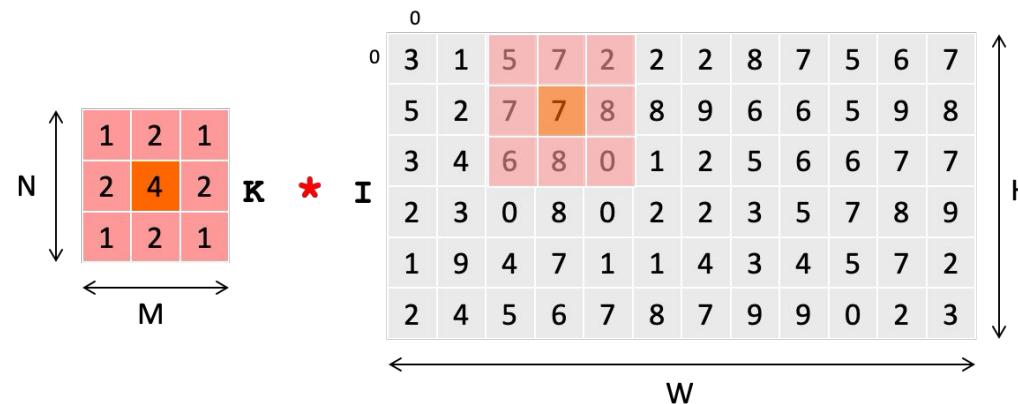
architecture hls of
function is
. . .
. . .
end hls;
```

RTL (VHDL, etc)

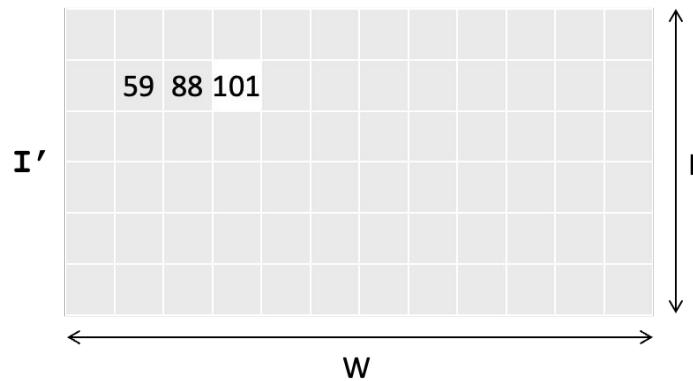
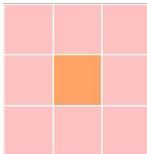


bitstream

Filtro di convoluzione con elaborazione *streaming* con FPGA



$$I'[1,3] = 1 \cdot 5 + 2 \cdot 7 + 1 \cdot 2 + 2 \cdot 7 + 4 \cdot 7 + 2 \cdot 8 + 1 \cdot 6 + 2 \cdot 8 + 1 \cdot 0 = 101$$



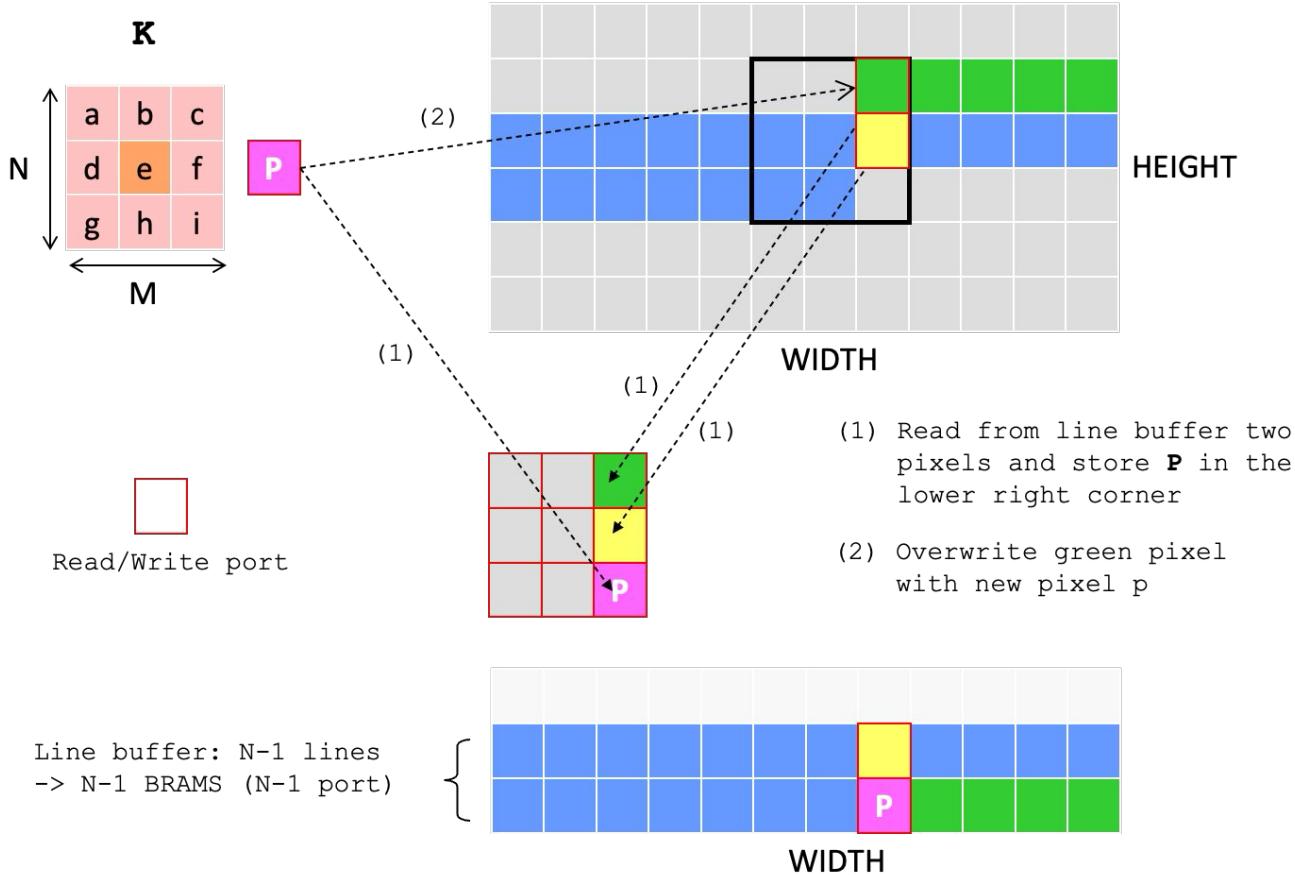
Struttura base codice HLS

- Utilizzando HLS, è possibile descrivere la convoluzione nel modo seguente
- Le #pragma servono per guidare il compilatore (C -> VHDL/Verilog)

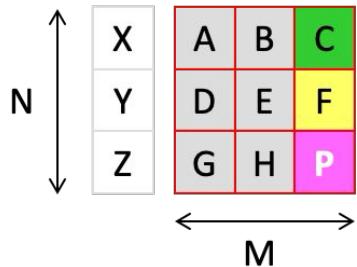
```
#include "ap_int.h"
typedef ap_uint<8> pixel;

void Filter(pixel input_img[640*480], pixel output_img[640*480])
{
    #pragma HLS INTERFACE axis port=out_img
    #pragma HLS INTERFACE axis port=in_img
    ...
    Loop_row: for (int row = 0; row < HEIGHT + (N-1)/2; row++)
        Loop_col: for int col = 0; col < WIDTH + (M-1)/2; col++)
            { #pragma HLS PIPELINE II=1
                // filter code here
            }
}
```

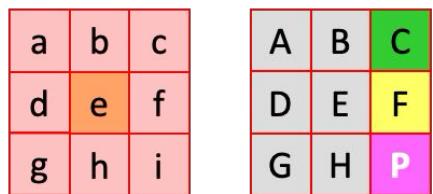
Data structures: line buffer (BRAM)



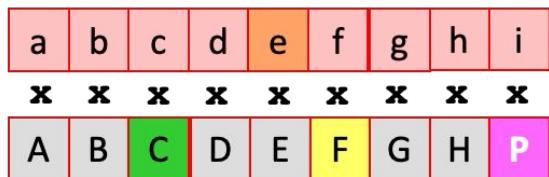
Data structures: image patch e kernel (FFD)



(3) The image patch is a shift register updated with the new N pixels (rightmost column)



(4) Dot product: $M \times N$ parallel read (K and patch).
For both data structures $M \times N$ read ports



$$aA + bB + cC + dD + eE + fF + gG + hH + iP$$