*Esercizio 12 – a scelta libera*

Progettare, implementare in VHDL e testare mediante simulazione (e, opzionalmente, mediante sintesi su board), un sistema le cui specifiche siano definite dallo studente e rientrino in una delle seguenti tipologie:

1. Modifica di esercizi già proposti (processore, rete di interconnessione o interfaccia seriale) mediante aggiunta/aggiornamento di funzionalità.

* Esempio: si potrebbe pensare di modificare l’interfaccia seriale aggiungendo segnali specifici per l’handshaking fra due entità.

1. Progetto di sistemi che assolvono a specifici compiti noti

* Esempio: si potrebbe pensare di implementare una specifica macchina aritmetica non trattata a lezione, una funzione crittografica, una rete neurale, ecc.

1. Progetto di sistemi che integrano opportunamente componenti visti a lezione (contatori, registri, macchine aritmetiche, ecc.).

* A titolo di esempio, è possibile fare riferimento ai seguenti due esercizi:

1. Progettare un sommatore di byte seriale (le cifre degli addendi devono essere fornite serialmente a coppie alla macchina) a partire da un sommatore di bit. Il sommatore deve terminare le sue operazioni appena il valore temporaneo della somma diventa maggiore di un valore M fornito in input.
2. Si consideri un nodo A che contiene una memoria ROM di N (N>=4) locazioni da 8 bit ciascuna. Progettare un sistema in grado di trasmettere mediante handshaking completo tutti i valori strettamente positivi contenuti nella memoria di A ad un nodo B. Il nodo B, ricevuti i valori da A, li trasmetterà ad un nodo C mediante una comunicazione parallela con handshaking.

Scelta progettuale:

E’ stato scelto di progettare ed implementare in VHDL un visualizzatore di immagini su schermo tramite periferica VGA ed il relativo protocollo di comunicazione. A partire dalla bitmap di un’immagine a colori, il sistema riesce a visualizzare l’immagine sullo schermo ed a muoverla all’interno dello spazio del display, andando a gestire le collisioni con i bordi. L’immagine è già presente in una memoria interna del sistema. Si utilizza come schermo di riferimento:

risoluzione in pixel 640x480

frequenza di aggiornamento dei frame 60Hz.

Realizzazione automatica della memoria:

A partire dall’immagine da mostrare si utilizza lo script Python “scriptcreazioneimmagine.py” per: convertire l’immagine rispetto al protocollo VGA, quindi 4 bit per ogni colore RGB; adattarla alla risoluzione in pixel dello schermo; generare automaticamente un file di testo che realizza la memoria interna del sistema. E’ stato opportuno effettuare un resize dell’immagine alle dimensioni 200x200 pixel, preservando le caratteristiche dell’immagine, in questo modo si evita di sovraccaricare il processo di sintesi ed implementazione su FPGA della memoria interna.

(Immagine iniziale)

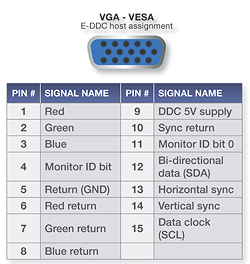
(Immagine elaborata)

Protocollo VGA:

Il protocollo VGA (Video Graphics Array) è uno standard di comunicazione con display per computer, prevede l’utilizzo di connettori DE-15 o anche detti VGA ed è uno standard analogico. Questo standard fornisce la risoluzione di 640x480 a 16 colori o di 320x400 a 256 colori. Ad oggi è stato esteso anche a risoluzioni più alte.

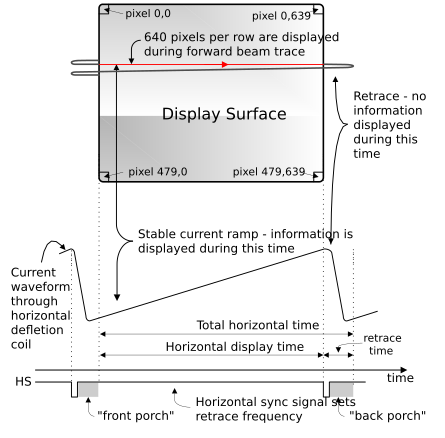
Il connettore VGA o DE-15 presenta 3 righe da 5 pin:

(Immagine pinout)



Ci si focalizza sui segnali in ingresso di: Red, Green e Blue, i quali indicano rispettivamente le intensità dei colori in modo analogico, dunque in base alla differenza di potenziale trasmessa; Horizontal sync e Vertical sync, i quali trasmettono un riferimento posizionale di dove bisogna mostrare il pixel.

Durante la visualizzazione lo schermo viene scansionato da sinistra verso destra e dal basso verso l’alto, procedendo riga per riga, andando a proiettare su schermo un fascio di luce di intensità e colore in base al segnale RGB analogico proveniente dal controller VGA. La deflessione del fascio di luce permette di far variare la posizione del pixel da mostrare, questo fenomeno viene controllato tramite i segnali di Sync. Il monitor VGA, a partire dai segnali di HSync e VSync, genera due forme d’onda (delle rampe con una certa frequenza) per la corrente che attraversa le bobine di deflessione, rispettivamente orizzontale e verticale. La frequenza di tali forme d’onda permette di descrivere la durata del tempo di display sia orizzontale che verticale, mentre l’ampiezza delle correnti ad ogni istante permette di definire il punto in cui visualizzare il pixel.

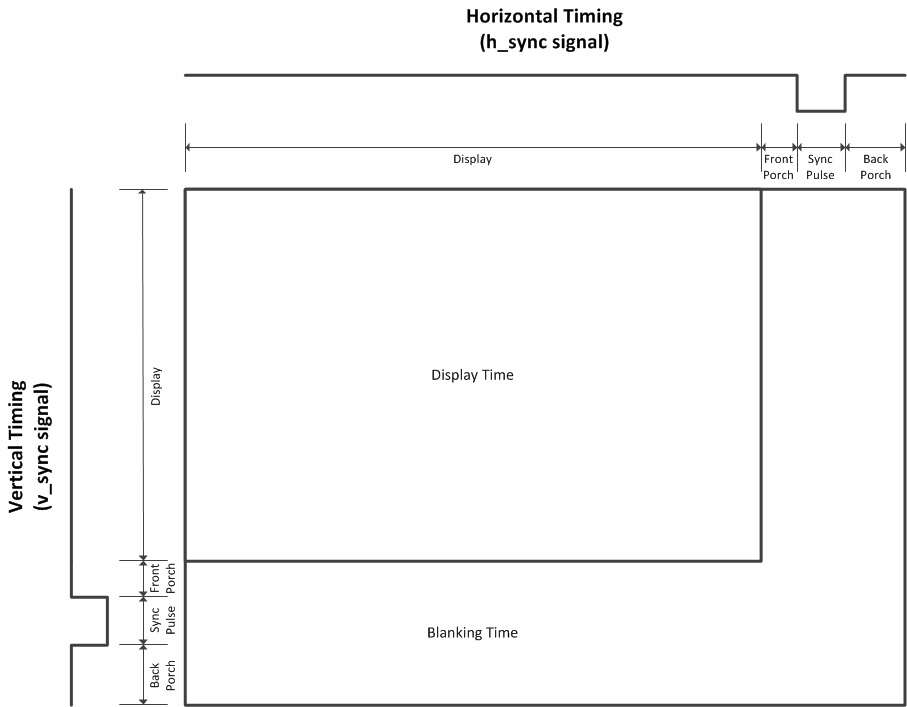


In quest’esempio viene mostrata solo la forma d’onda orizzontale, a man a mano che il valore della corrente aumenta ci si sposta verso destra nel display, fino a raggiungere la fine dello schermo. Qui inizia la fase di “blanking” orizzontale, in cui il fascio di luce viene resettato per poter iniziare una nuova scansione sulla riga successiva e non viene mostrato nulla a schermo. La durata della fase di blanking orizzontale è data da: Front Porch + sync pulse + back porch.

La fase di “blanking” si ripete anche per la scansione in verticale: una volta terminate le righe da mostrare, bisogna ritornare alla riga iniziale dello schermo e resettare ancora una volta il fascio di luce prodotto. La durata della fase di “blanking” verticale ha la stessa formula di quella orizzontale.

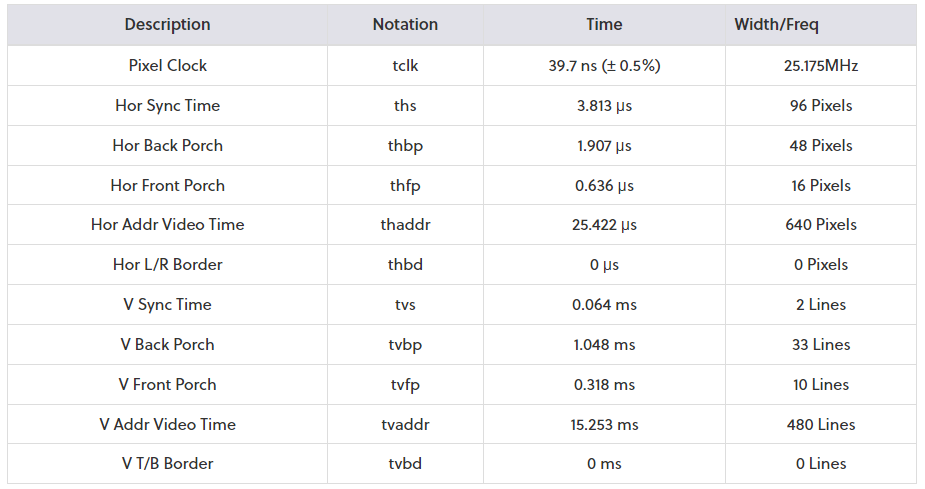
I tempi di Front Porch, sync pulse e back porch, sia orizzontali che verticali, vengono calcolata in base alla risoluzione dello schermo.

Il segnale di HS e di VS saranno quindi formati da una regione attiva, in cui vengono estratti dalla memoria i pixel da mostrare, ed una regione di blanking per sincronizzare la visualizzazione, in cui i pixel non vengono estratti e si disattiva un segnale di abilitazione del display.



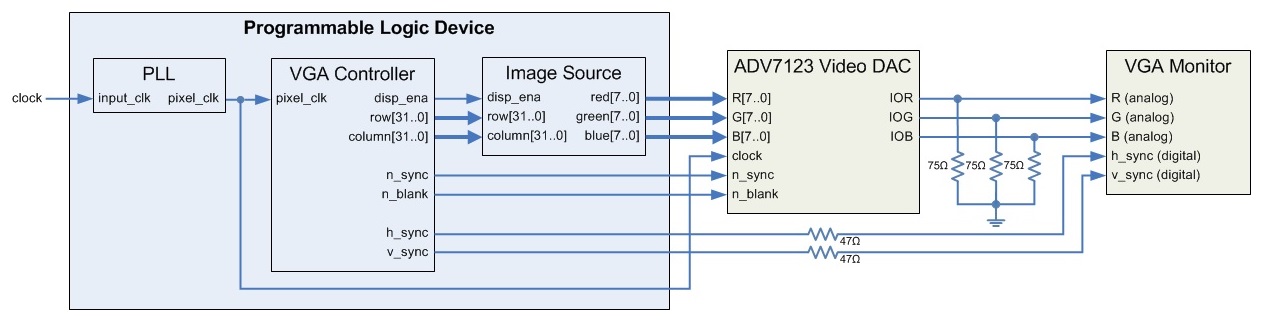
E’ necessario sincronizzare i dati video trasmessi allo schermo con la posizione alla quale vogliono essere visualizzati, ossia sincronizzare la consegna dei dati video allo schermo con i segnali di HS e VS. Ciò avviene tramite un pixel clock, che definisce la frequenza con la quale viene visualizzato un singolo pixel di informazione. Ad ogni colpo di pixel clock, si incrementano il contatore orizzontale e verticale (seguendo la scansione riga per riga) : in base al valore che si ottiene dai contatori si stabilisce se ci si trova nella regione di visualizzazione o in quella di blanking, nel primo caso si indicizza in memoria al valore dei contatori per poter estrarre il pixel da mostrare. Al colpo di pixel clock successivo il dato viene portato in ingresso allo schermo e mostrato in uscita. In base al valore dei contatori, si passa da una fase all’altra generando i sync pulse.

Per uno schermo 640x480 con frequenza di aggiornamento pari a 60Hz, le specifiche temporali sono:

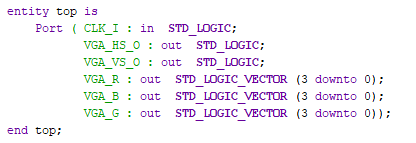


Implementazione del controller VGA:

(schema controller VGA)



Il controller in ingresso riceve il clock della board e produce in uscita i segnali: Red, Green e Blue codificati ognuno su 4 bit; H\_sync e V\_sync. I primi 3 vengono resi analogici dalla board in automatico tramite un DAC.

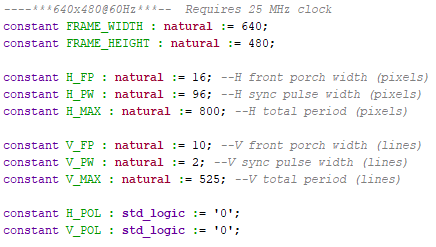


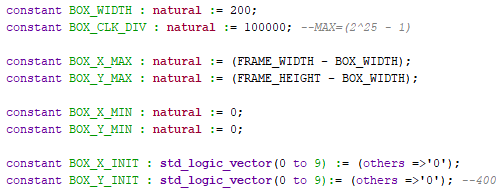
Per la realizzazione del controller VGA è stata utilizzata un progetto demo della Digilent per l’utilizzo della VGA, relativo alle board Artyx.( <https://digilent.com/reference/learn/programmable-logic/tutorials/arty-pmod-vga-demo/start>)

E’ stato rimodellato il top module, ossia quello che realizza il controller VGA, per farsi che: venga prelevato dalla memoria il pixel da mostrare ad ogni colpo di pixel clock; vengano gestite le collisioni tra l’immagine ed i bordi. Il tutto utilizzando un approccio misto tra comportamentale e strutturale.

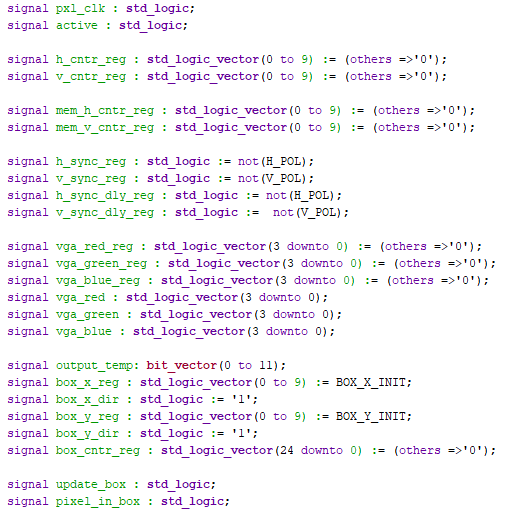
Il sistema complessivo è formato dal controller VGA che comunica con una memoria che contiene l’immagine, il tutto sincronizzato tramite un pixel clock realizzato come divisore in frequenza tramite il Clock Wizard presente nell’Ip Catalog.

Vengono dichiarate le costanti relative alla tempificazione:

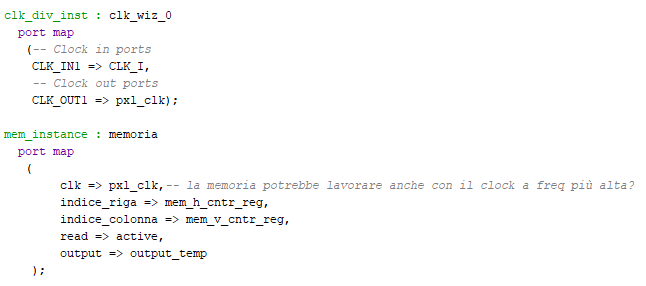


Vengono dichiarate le costanti relative al box dell’immagine::

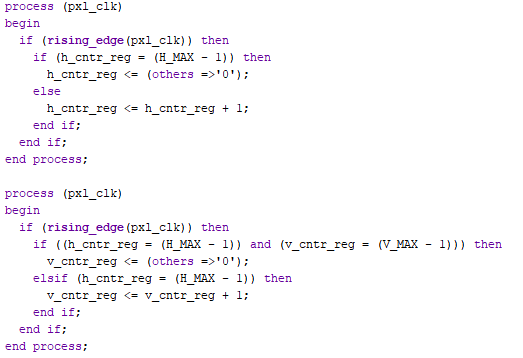
Vengono dichiarati i segnali che interagiscono internamente al controller:



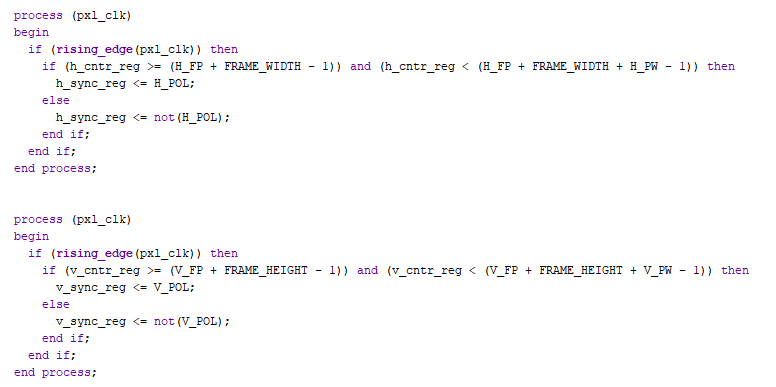
I componenti interni al top module sono:



Vengono, poi, definiti due contatori in cascata, in modo comportamentale, di modulo rispettivamente pari a: H\_MAX e V\_MAX. Questi permettono l’indicizzazione in memoria del pixel successivo a quello che si sta visualizzando tramite i segnali h\_cntr\_reg e v\_cntr\_reg.



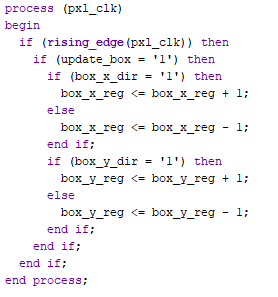
In base al valore dei contatori, orizzontale e verticale, vengono generati i segnali di h\_sync e v\_sync tramite i seguenti process:



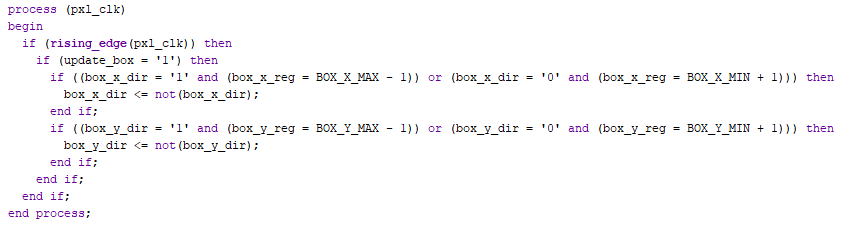
In entrambi i process, la prima condizione indica se il contatore si trova nella fase di blanking, in questo caso allora genera il sync pulse della durata H\_PW o V\_PW, altrimenti il segnale di sync rimane alto.

Per la gestione del box dell’immagine e il relativo movimento, sono stati creati 3 process, in cui si utilizza il primo pixel in alto a sinistra del box come posizione di riferimento per il box totale. Al box viene, poi assegnata una direzione corrente sia orizzontale che verticale, che indica dove si sta muovendo.

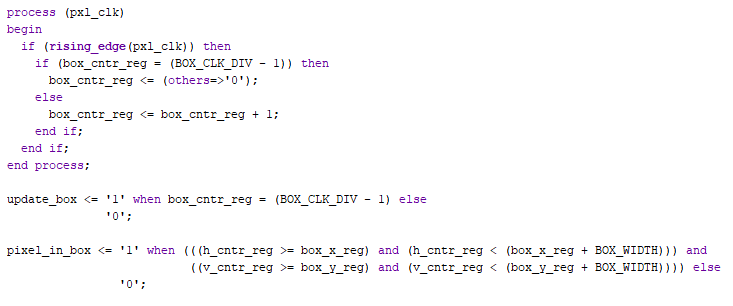
-il primo gestisce il movimento in base alla direzione corrente (ad esempio se si sposta in diagonale verso il basso, allora incremento di 1 la posizione del box in entrambe le direzioni x e y);



-Il secondo gestisce le collisioni dell’immagine sui bordi dello schermo e quindi fa variare la direzione corrente del box, ad esempio se la posizione del box raggiunge il limite inferiore dello schermo, allora devo invertire la direzione corrente verticale y.



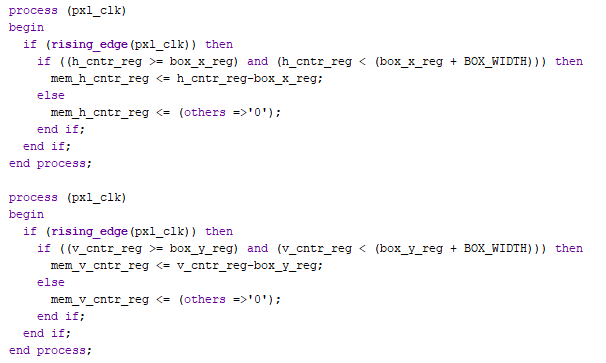
-il terzo realizza un divisore in frequenza, che gestisce ogni quando viene effettuato l’update del box su schermo, cioè quante volte si entra all’interno dei process precedenti.



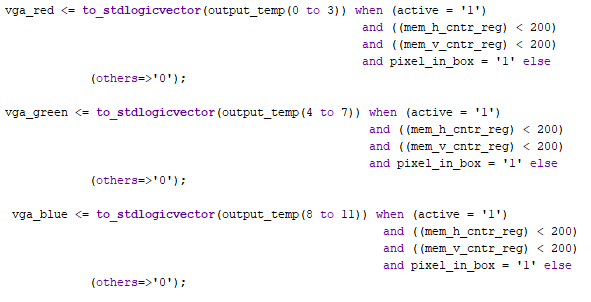
Il segnale pixel\_in\_box indica se il pixel corrente, cioè quello puntato dal contatore orizzontale e verticale, si trova all’interno del box contenente l’immagine.

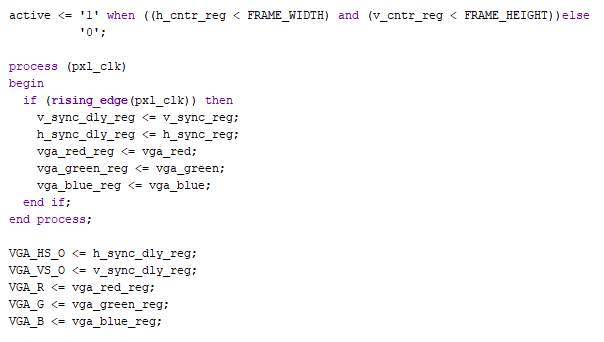
Per aumentare la velocità del box basta diminuire la divisione in frequenza, più è basso il valore di BOX\_CLK\_DIV maggiori saranno i numeri di update e dunque maggiore sarà la velocità del box.

Per accedere in memoria si utilizzano due registri: mem\_h\_cntr\_reg, mem\_v\_cntr\_reg. Questi vengono assegnati a partire dai valori di h\_cntr\_reg e v\_cntr\_reg, a cui vengono sottratti rispettivamente box\_x\_reg e box\_y\_reg, ossia la posizione del box. Ciò avviene solo se il pixel si trova nel box.



Si assegnano poi i segnali di Red, Green e Blue con quelli in uscita dalla memoria solo se si è in regione attiva ed il pixel appartiene al box, altrimenti li si pone a 0.

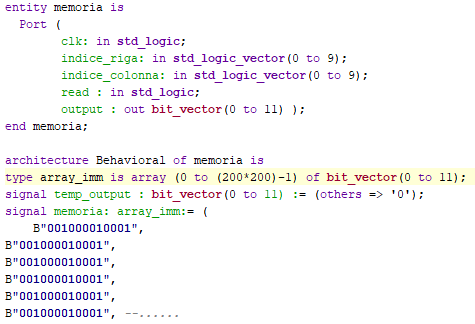




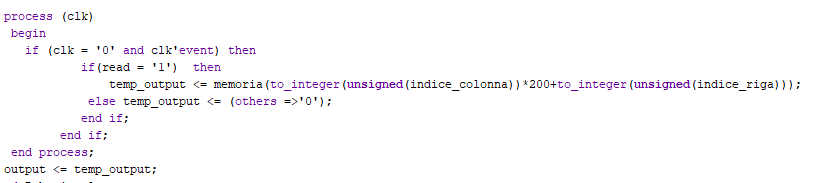
Si sincronizzano tutte le uscite del controller VGA, rispetto al pixel clock, tramite dei registri intermedi.

Memoria:

La memoria è di sola lettura ed è realizzata come array di BIT\_VECTOR, ognuno di 12 elementi: 4 bit per il rosso, 4 bit per il verde, 4 bit per il blu.



Per leggere il valore all’indirizzo dato da (indice\_riga, indice\_colonna) si utilizza il process:



Dove l’indicizzazione è in questo modo perché l’immagine viene inserita come matrice serializzata lungo le colonne.

Esempi di risultati ottenuti:

(Visualizzazione su schermo dell’immagine che si muove)