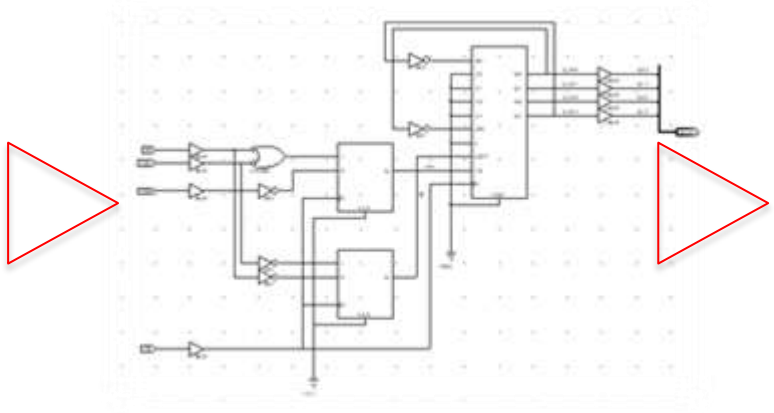
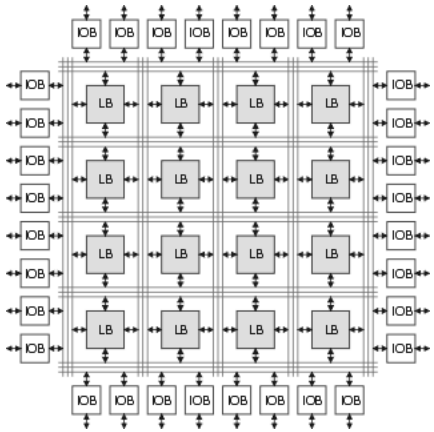
ALMA MATER STUDIORUM – UNIVERSITY OF BOLOGNA
DEIS - DEPARTMENT OF ELECTRONICS, COMPUTER ENGINEERING AND SYSTEMS

```
begin
  if (RESET_N = '0') then

    for col in 0 to BOARD_COLUMNS-1 loop
    for row in 0 to BOARD_ROWS-1 loop
    ...
   elsif (rising_edge(CLOCK)) then
    ...
```





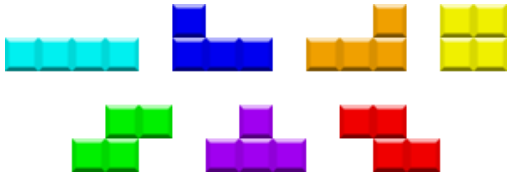
Laboratorio di Sistemi Digitali M
A.A. 2010/11

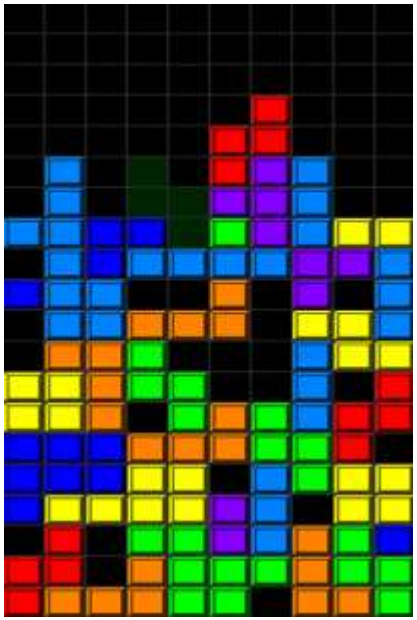
3 – Esercitazione Tetris:
Introduzione ed architettura

Primiano Tucci
primiano.tucci@unibo.it

www.primianotucci.com

Definizione del problema





Board: Una scacchiera di dimensioni arbitrarie
Piece: 7 pezzi base (STICK, L_R, L_L, SQUARE, DOG_R, T, DOG_L)

I pezzi, scelti casualmente dal sistema, "cadono" uno per volta nella scacchiera fino a raggiungere una posizione stabile, conformemente alla loro geometria e compatibilmente con i pezzi già presenti nella scacchiera (inizialmente vuota).

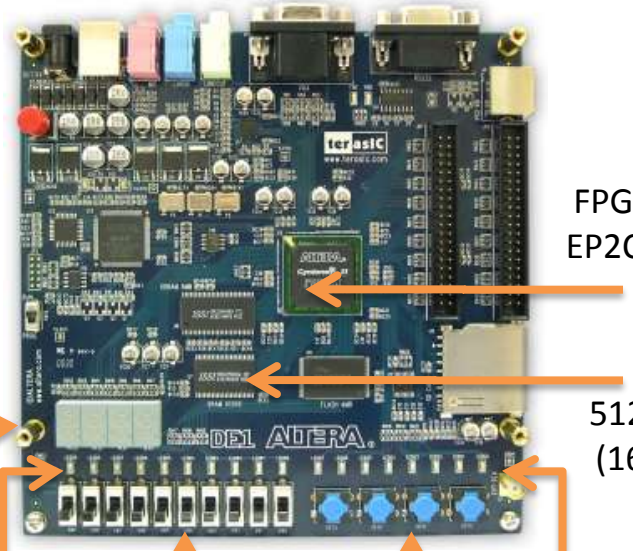
Durante la loro discesa il giocatore, per mezzo di opportuni comandi, può spostare il pezzo in orizzontale, ruotarlo (compatibilmente con la presenza di altri pezzi nella scacchiera) o accelerarne la discesa.

Qualora un pezzo raggiunga una posizione "gravitazionalmente stabile" e concorra al completamento di una o più righe della scacchiera, causa il cancellamento delle suddette righe e la conseguente "caduta" delle eventuali righe superiori.

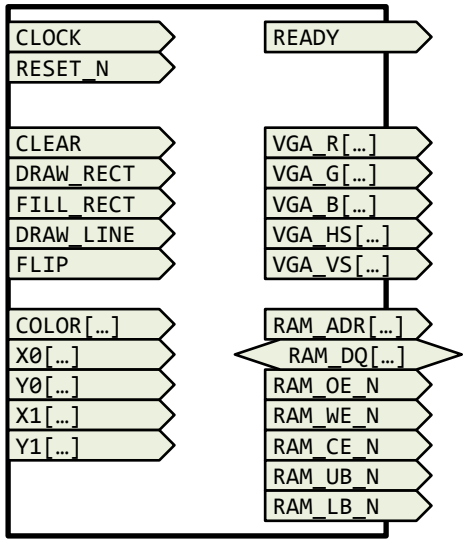
La partita termina quando non vi è più spazio sufficiente per consentire la caduta di ulteriori pezzi.

Cosa abbiamo a disposizione

Schede dimostrative Terasic-Altera DE1



Un motore grafico + controller VGA



Tanti eccellenti designer hardware!

Obiettivi

- Applicare principi di design
- Approfondire conoscenza del linguaggio VHDL
- Acquisire dimestichezza con hardware reale
- Lavoro di squadra all'interno di un progetto Open Source

Requisiti

- Altera Quartus II Web Edition (9 o sup.)
  Scaricabile gratuitamente presso https://www.altera.com/download/software/quartus-ii-we
  NON sono necessari Modelsim e NIOS-II

- Un account google (NON necessariamente GMail, es. collegato all'indirizzo istituzionale)
  Seguire le istruzioni su http://code.google.com/p/tetris-vhdl/wiki/Istruzioni

- **Partecipazione attiva**
  Siete *fortemente invitati* a partecipare durante la lezione ed a discutere tra di voi durante lo sviluppo sull'apposito forum del progetto http://code.google.com/p/tetris-vhdl/

Organizzazione

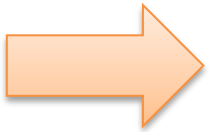
- Indirizzo e-mail: primiano.tucci@unibo.it
- Cortesemente, fate iniziare l'oggetto di tutte le mail che inviate mail per "[LABSD11] oggetto"
- E' necessario definire 6 team**.** Inviatemi una mail per team, contenente i nomi ed e-mail dei membri.
- **Fase iniziale (3/4 settimane)**
  - Porteremo avanti insieme lo sviluppo iniziale. Ci daremo degli obiettivi settimanali, comuni a tutti i team.
  - Prima di ogni lezione commenteremo e confronteremo le soluzioni che avete trovato durante la settimana.
  - Porteremo avanti lo sviluppo in modo uniforme, fino ad arrivare ad una *base* funzionante e condivisa.
- **Progetti**
  - Ad ogni team verranno assegnate delle estensioni al progetto su cui lavorare.

- Sito del progetto (open source): http://code.google.com/p/tetris-vhdl
- Discussioni progetto : http://groups.google.com/group/tetris-vhdl-dev
- Discussioni/dubbi/richieste VHDL: http://groups.google.com/group/it-comp-lang-vhdl

Architettura generale

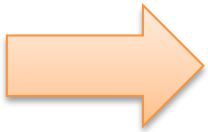
Ricordi di Ing. del Software L-A : Il pattern MVC (a volte tornano)

- **Model**
  Gestisce un insieme di dati logicamente correlati
  Risponde alle interrogazioni sui dati
  Risponde alle istruzioni di modifica dello stato



Datapath

- **Controller**
  Gestisce gli input dell'utente
  Mappa le azioni dell'utente in comandi
  Invia tali comandi al modello e/o alla view che effettuano le operazioni appropriate

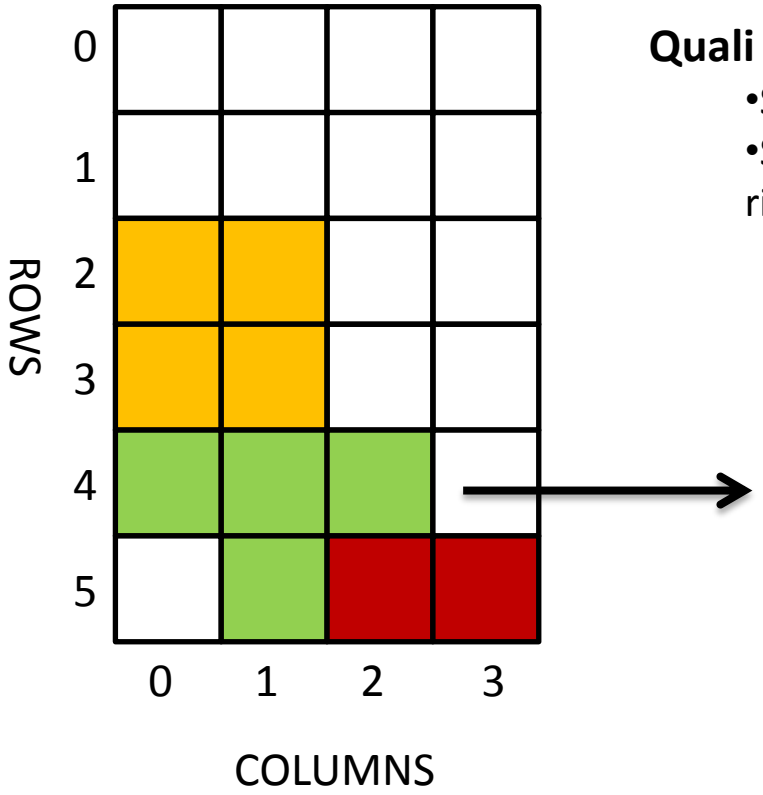


Control Unit

- **View**
  Gestisce un'area di visualizzazione, nella quale presenta all'utente una vista dei dati gestiti dal modello
  Mappa i dati (o parte dei dati) del modello in oggetti visuali. Visualizza tali oggetti su un (particolare) dispositivo di output

View

Scacchiera (*board*)



Quali informazioni veicola ogni cella della *board*?

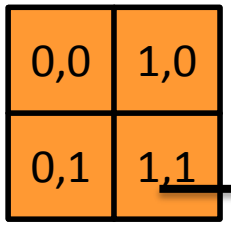
- Se è vuota oppure no!
- Se non è vuota: la *tipologia* di pezzo che ha contribuito a riempirla

```
Cell[R,C]
• filled (1/0)
• shape ({DOG, T, STICK…})
```

Pezzi (*piece*)

Quali informazioni veicola un pezzo?

- La tipologia (da cui deriveranno proprietà secondarie, es. il colore)
- La morfologia: coordinate di ogni suo blocco



Hint: siamo veramente fortunati, tutti i pezzi del tetris sono composti dallo stesso numero (4) di blocchi elementari.

`Piece`

- `shape`
- `blocks[0..3]`
  - `col (integer 0 to COLS-1)`
  - `row (integer 0 to ROWS-1)`

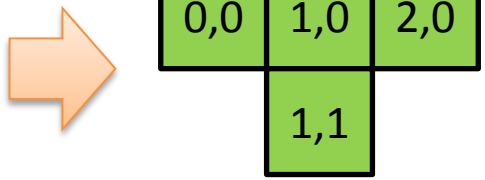
In VHDL: il tetris_package

```
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;
use work.vga_package.all;
```

```
package tetris_package is
    constant  BOARD_COLUMNS    : positive   := 10;
    constant  BOARD_ROWS       : positive   := 20;
    constant  BLOCKS_PER_PIECE : positive   := 4;

    type      shape_type     is (SHAPE_T, SHAPE_SQUARE, SHAPE_STICK, SHAPE_L_L, SHAPE_L_R, SHAPE_DOG_L, SHAPE_DOG_R);
    attribute enum_encoding  : string;
    attribute enum_encoding  of shape_type : type is "one-hot";

    type board_cell_type is record
            filled       : std_logic;
            shape        : shape_type;
    end record;

    type board_cell_array is array(natural range <>, natural range <>) of board_cell_type;

    type board_type is record
            cells        :  board_cell_array(0 to (BOARD_COLUMNS-1), 0 to (BOARD_ROWS-1));
    end record;

    type block_pos_type is record
            col         : integer range 0 to (BOARD_COLUMNS-1);
            row         : integer range 0 to (BOARD_ROWS-1);
    end record;

    type block_pos_array is array(natural range <>) of block_pos_type;

    type piece_type is record
            shape        : shape_type;
            blocks       : block_pos_array(0 to (BLOCKS_PER_PIECE-1));
    end record;
```

Dai concetti a VHDL (continua)

```
    -- Piece definitions
    constant PIECE_T : piece_type :=
    (
            shape  => SHAPE_T,
            blocks =>
            (
                        (col => 0, row => 0),
                        (col => 1, row => 0),
                        (col => 2, row => 0),
                        (col => 1, row => 1)
            )
    );

    constant PIECE_SQUARE: piece_type := OMISSIS
    constant PIECE_STICK : piece_type := OMISSIS
    constant PIECE_L     : piece_type := OMISSIS
    constant PIECE_LR    : piece_type := OMISSIS
    constant PIECE_DOG_L : piece_type := OMISSIS
    constant PIECE_DOG_R : piece_type := OMISSIS

    function Lookup_color(shape : shape_type) return color_type;
end package;
```



Dai concetti a VHDL (continua)

```
package body tetris_package is

    function Lookup_color(shape : shape_type)
            return color_type is
            variable color : color_type;
    begin
            case (shape) is
                    when SHAPE_T =>
                            color := COLOR_YELLOW;
                    when SHAPE_SQUARE =>
                            color := COLOR_MAGENTA;
                    when SHAPE_STICK =>
                            color := COLOR_ORANGE;
                    when SHAPE_L_L | SHAPE_L_R =>
                            color := COLOR_CYAN;
                    when SHAPE_DOG_R | SHAPE_DOG_L =>
                            color := COLOR_GREEN;
                    when others =>
                            color := COLOR_WHITE;
            end case;
            return color;
    end function;

end package body;
```

Abbiamo definito i tipi di dato che useremo!
Per ora, però, non abbiamo definito neanche un gate!

Per la settimana prossima

- Tutti i TEAM: definizione del datapath. Ovvero

A partire dai tipi di dato che abbiamo definito
(il file vhdl contente il package è sul sito del progetto)

1. Identificazione elementi di memoria (registri).
2. Identificazione operazioni sui registri
3. Definizione black-box datapath e segnali scambiati con Control Unit e View
4. Definizione RTL
5. Provate a dare una implementazione in VHDL del tutto