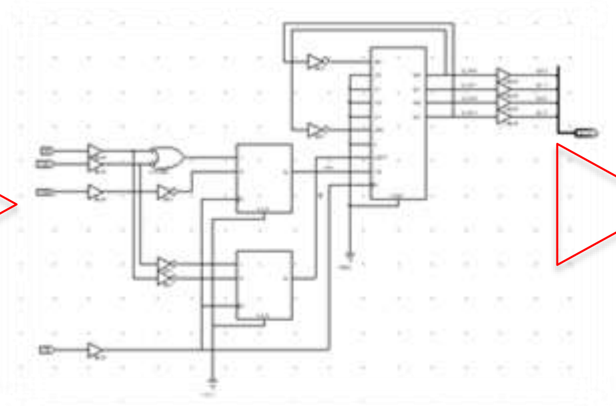
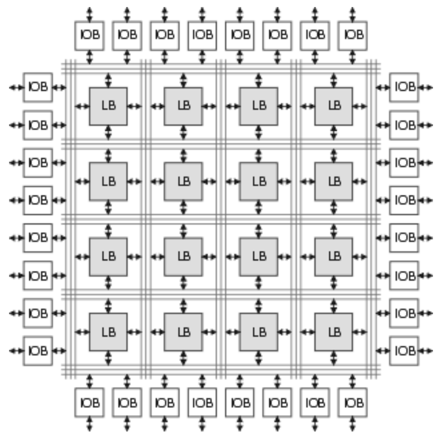
```
begin
  if (RESET_N = '0') then

    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

6 – Esercitazione Tetris: View

Primiano Tucci
primiano.tucci@unibo.it

www.primianotucci.com

Agenda

- Il view deve compiere una sola macro-operazione:
  a fronte della richiesta REDRAW, deve ridisegnare la scena.
- Le micro-operazioni da compiere sono, in sequenza:
(1) CLEAR del framebuffer; (2) disegno del perimetro della board; (3) disegno dei singoli blocchi della board (ove presenti); (4) FLIP del framebuffer
- Ogni operazione sul framebuffer può essere eseguita previa verifica del segnale READY
- Assumete di avere abbastanza tempo per disegnare la board, ovvero, tra quando viene asserito il segnale REDRAW e quando finite di disegnare la board, il contenuto della board rimane invariato.

Segnali Controller→View

REDRAW

Segnali Datapath ←→View

QUERY_CELL[…]

CELL_CONTENT[…]

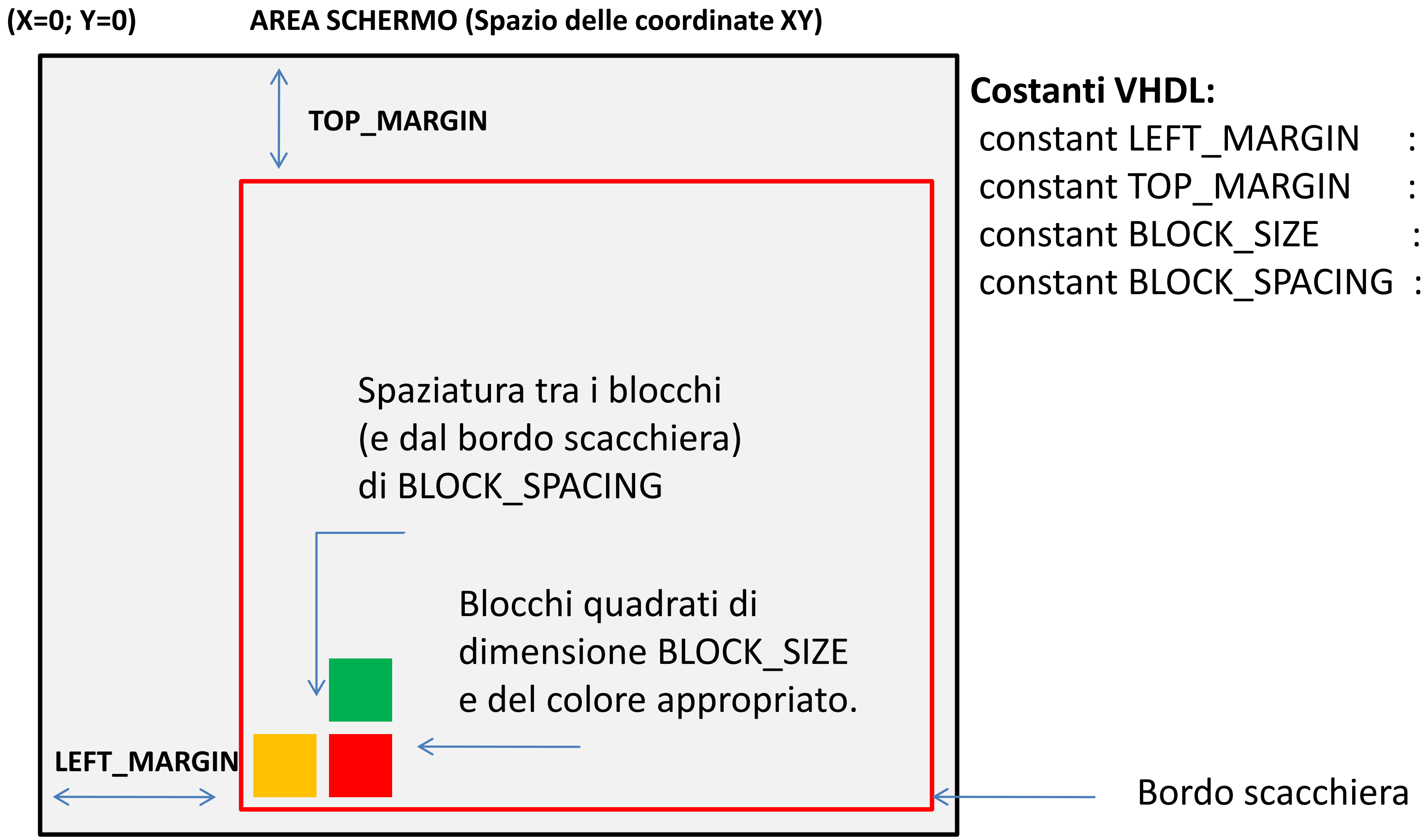
Segnali Framebuffer VGA←→View

READY

CLEAR
DRAW_RECT
FILL_RECT
DRAW_LINE
FLIP

COLOR[…]
X0[…]
Y0[…]
X1[…]
Y1[…]

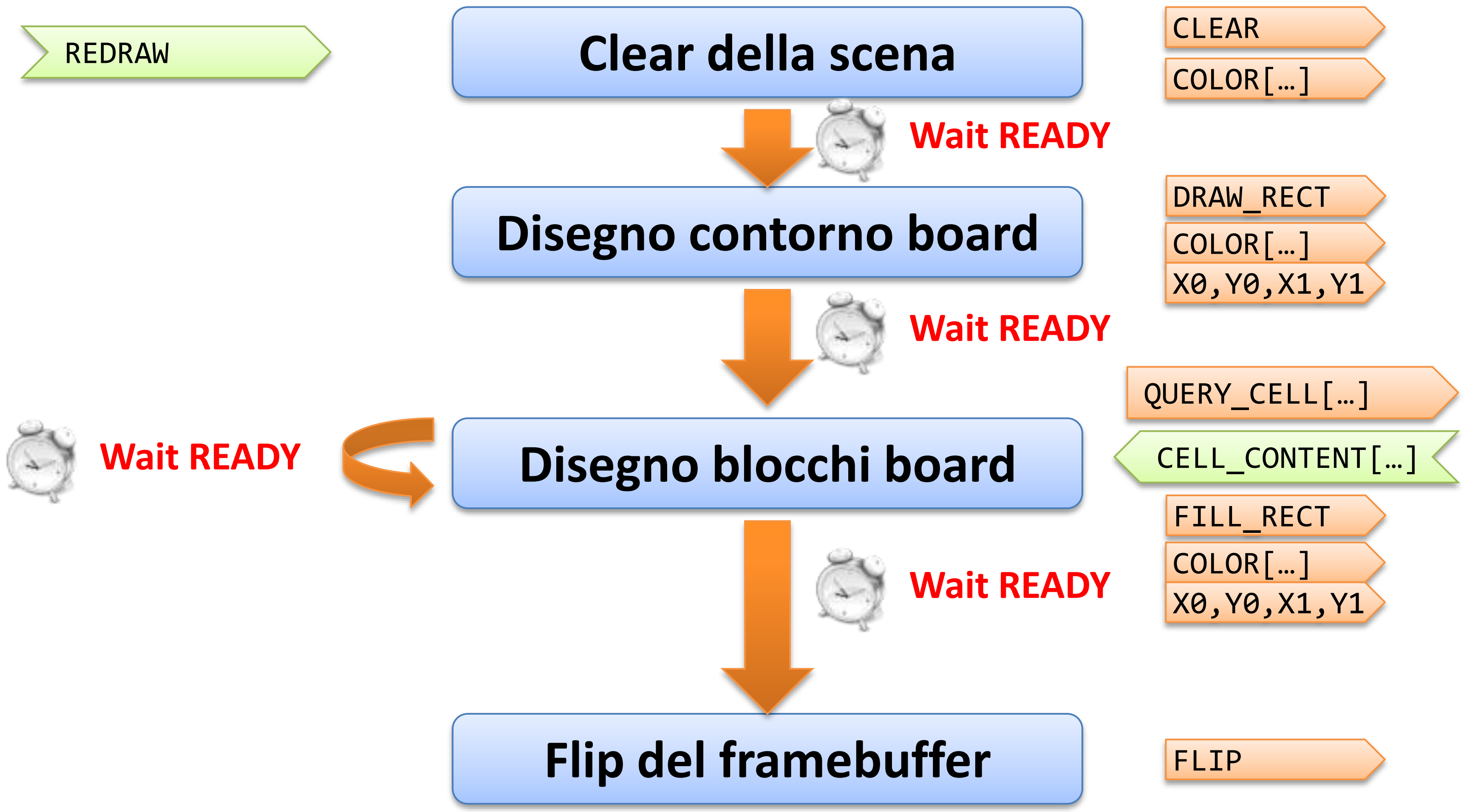
Specifiche per il disegno della board



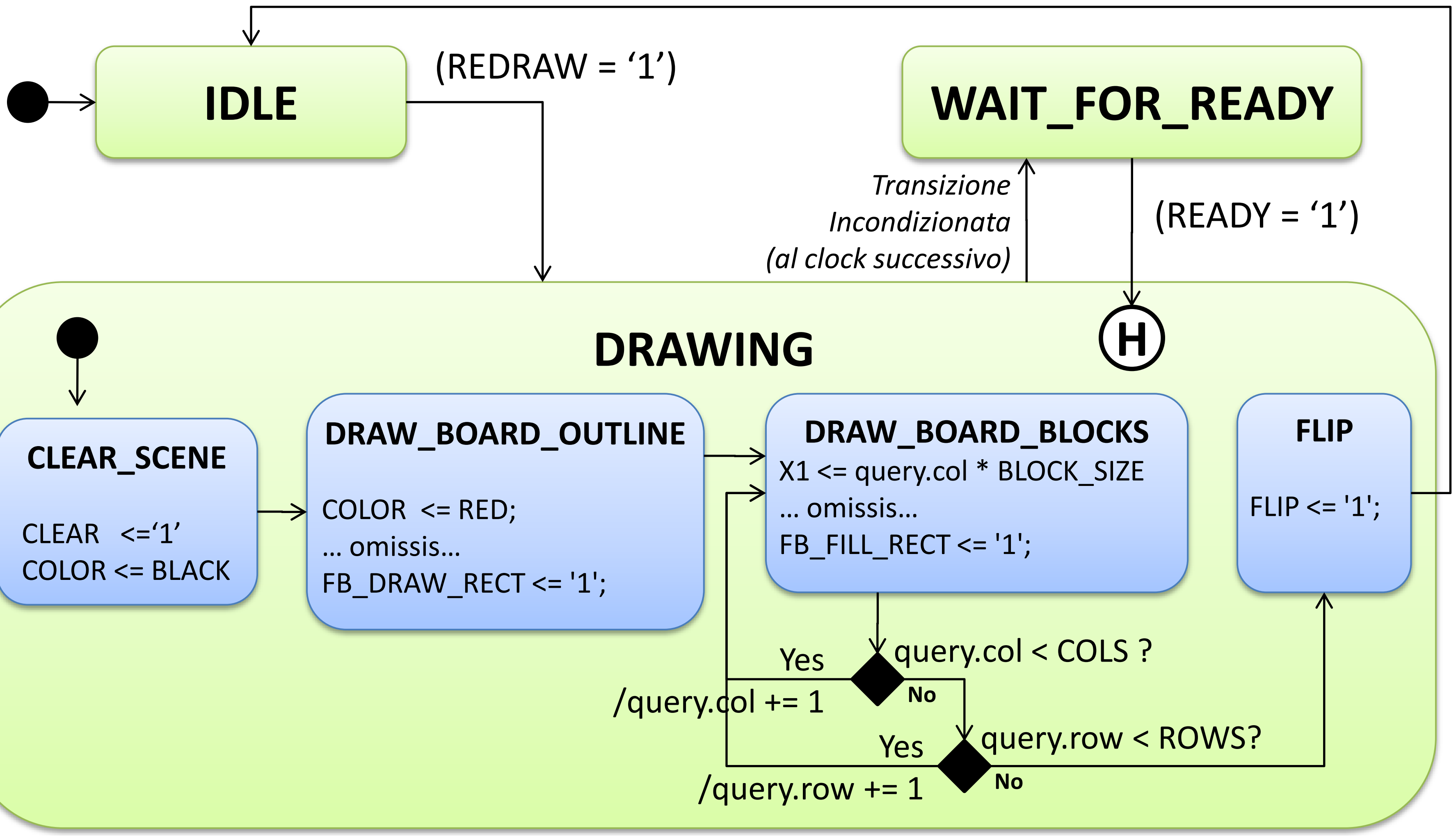
Costanti VHDL:

```
constant LEFT_MARGIN     : integer
constant TOP_MARGIN      : integer;
constant BLOCK_SIZE      : integer;
constant BLOCK_SPACING   : integer;
```

Obiettivo


REDRAW
Clear della scena
CLEAR
COLOR[…]
Wait READY
Disegno contorno board
DRAW_RECT
COLOR[…]
X0,Y0,X1,Y1
Wait READY
QUERY_CELL[…]
Wait READY
Disegno blocchi board
CELL_CONTENT[…]
FILL_RECT
COLOR[…]
X0,Y0,X1,Y1
Wait READY
Flip del framebuffer
FLIP

State Chart


IDLE
(REDRAW = '1')
WAIT_FOR_READY
Transizione Incondizionata (al clock successivo)
(READY = '1')
DRAWING
H
CLEAR_SCENE
CLEAR <='1'
COLOR <= BLACK
DRAW_BOARD_OUTLINE
COLOR <= RED;
… omissis…
FB_DRAW_RECT <= '1';
DRAW_BOARD_BLOCKS
X1 <= query.col * BLOCK_SIZE
… omissis…
FB_FILL_RECT <= '1';
FLIP
FLIP <= '1';
query.col < COLS ?
Yes
/query.col += 1
No
query.row < ROWS?
Yes
/query.row += 1
No

VHDL Entità View

```
entity Tetris_View is
    port
    (
          CLOCK           : in  std_logic;
          RESET_N         : in  std_logic;

          REDRAW          : in  std_logic;

          FB_READY        : in  std_logic;
          FB_CLEAR        : out std_logic;
          FB_DRAW_RECT    : out std_logic;
          FB_DRAW_LINE    : out std_logic;
          FB_FILL_RECT    : out std_logic;
          FB_FLIP         : out std_logic;
          FB_COLOR        : out color_type;
          FB_X0           : out xy_coord_type;
          FB_Y0           : out xy_coord_type;
          FB_X1           : out xy_coord_type;
          FB_Y1           : out xy_coord_type;

          QUERY_CELL      : out block_pos_type;
          CELL_CONTENT    : in  board_cell_type
    );
end entity;
```

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

VHDL Architettura (1/4)

```
architecture RTL of Tetris_View is
   constant LEFT_MARGIN    : integer := 8;  -- Margine tra board e schermo
   constant TOP_MARGIN     : integer := 8;   -- IDEM (margine alto)
   constant BLOCK_SIZE     : integer := 20; -- Dimensione singolo blocco
   constant BLOCK_SPACING  : integer := 1; -- Dimensione singolo blocco


   type   state_type    is (IDLE, WAIT_FOR_READY, DRAWING);
   type   substate_type is
        (CLEAR_SCENE, DRAW_BOARD_OUTLINE, DRAW_BOARD_BLOCKS, FLIP_FRAMEBUFFER);
   signal state        : state_type;
   signal substate     : substate_type;
   signal query_cell_r : block_pos_type;

begin
   QUERY_CELL <= query_cell_r;
```

VHDL Architettura (2/4)

```
process(CLOCK, RESET_N)
   begin
        if (RESET_N = '0') then
                state            <= IDLE;
                substate         <= CLEAR_SCENE;
                FB_CLEAR         <= '0';
                FB_DRAW_RECT     <= '0';
                FB_DRAW_LINE     <= '0';
                FB_FILL_RECT     <= '0';
                FB_FLIP          <= '0';
                query_cell_r.col <= 0;
                query_cell_r.row <= 0;

        elsif (rising_edge(CLOCK)) then
                FB_CLEAR      <= '0';
                FB_DRAW_RECT  <= '0';
                FB_DRAW_LINE  <= '0';
                FB_FILL_RECT  <= '0';
                FB_FLIP       <= '0';
```

Uscite attive solo per un periodo di clock, quando “smentite” (più sotto)

VHDL Architettura (3/4)

```
case (state) is
  when IDLE =>
    if (REDRAW = '1') then
      state    <= DRAWING;
      substate <= CLEAR_SCENE;
    end if;

  when WAIT_FOR_READY =>
    if (FB_READY = '1') then
      state <= DRAWING;
    end if;

  when DRAWING =>
    state <= WAIT_FOR_READY;
```

Ogni operazione grafica fatta nello stato DRAWING per default transita (al clock successivo) in WAIT_FOR_READY

```
case (substate) is
    when CLEAR_SCENE =>
      FB_COLOR      <= COLOR_BLACK;
      FB_CLEAR      <= '1';
      substate      <= DRAW_BOARD_OUTLINE;

    when DRAW_BOARD_OUTLINE =>
      FB_COLOR      <= COLOR_RED;
      FB_X0         <= LEFT_MARGIN;
      FB_Y0         <= TOP_MARGIN;
      FB_X1         <= LEFT_MARGIN +
      (BOARD_COLUMNS * BLOCK_SIZE);
      FB_Y1         <= TOP_MARGIN  + (BOARD_ROWS *
      BLOCK_SIZE);
      FB_DRAW_RECT <= '1';
      substate      <= DRAW_BOARD_BLOCKS;

    when DRAW_BOARD_BLOCKS => Slide successiva
    when FLIP_FRAMEBUFFER =>
      FB_FLIP  <= '1';
      state    <= IDLE;
```

VHDL Architettura (4/4)

```
when DRAW_BOARD_BLOCKS =>
   if(CELL_CONTENT.filled = '1') then
     FB_COLOR   <= Lookup_color(CELL_CONTENT.shape);
     FB_X0   <= LEFT_MARGIN + (query_cell_r.col * BLOCK_SIZE) + BLOCK_MARGIN;
     FB_Y0   <= TOP_MARGIN  + (query_cell_r.row * BLOCK_SIZE) + BLOCK_MARGIN;
     FB_X1   <= LEFT_MARGIN + (query_cell_r.col * BLOCK_SIZE) + BLOCK_SIZE - BLOCK_MARGIN;
     FB_Y1   <= TOP_MARGIN  + (query_cell_r.row * BLOCK_SIZE) + BLOCK_SIZE - BLOCK_MARGIN;
     FB_FILL_RECT <= '1';
   end if;

   if (query_cell_r.col /= BOARD_COLUMNS-1) then
     query_cell_r.col <= query_cell_r.col + 1;
   else
     query_cell_r.col <= 0;
     if (query_cell_r.row /= BOARD_ROWS-1) then
         query_cell_r.row <= query_cell_r.row + 1;
     else
         query_cell_r.row <= 0;
         substate  <= FLIP_FRAMEBUFFER;
     end if;
   end if;
```