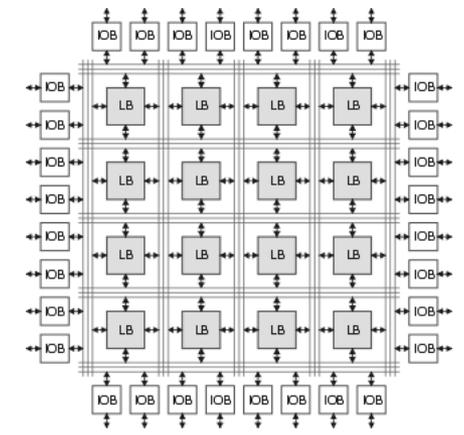
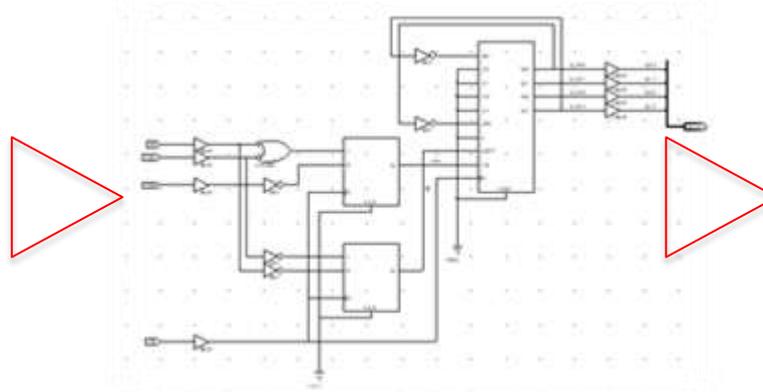


```
begin  
  if (RESET_N = '0') then  
    for col in 0 to BOARD_COLUMNS-1 loop  
      for row in 0 to BOARD_ROWS-1 loop  
        ...  
      elsif (rising_edge(CLOCK)) then  
        ...  
      end loop  
    end loop  
  end if  
end
```



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



5 – Esercitazione Tetris: Control Unit

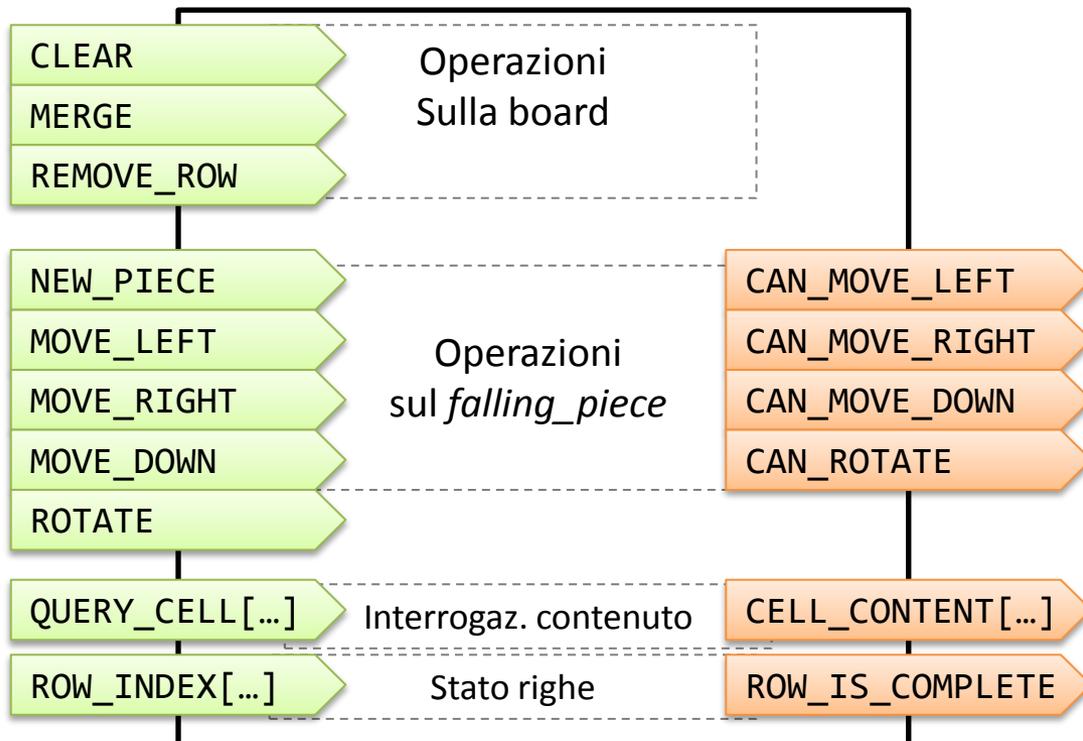
Primiano Tucci
primiano.tucci@unibo.it

www.primianotucci.com

Agenda

1. Definizione operazioni
2. Modalità di interazione con il Datapath e View
3. Implementazione in VHDL

Precedentemente abbiamo modellato il datapath come segue:



Attenzione alla modalità di interazione con i servizi

Ogni servizio viene espletato quando, al fronte del clock, il relativo segnale è asserito.

Se il segnale rimane asserito per due cicli di clock, il servizio viene eseguito due volte!



Modellazione di servizi: tipologia

COMBINATORIO:

Il servizio in realtà è una semplice funzione combinatoria di uno o più input e non altera lo stato interno della entità.

Es: segnale IS_EMPTY di una struttura dati

ONE-SHOT:

Non è possibile revocare o interrompere un servizio . L'interruzione è fisicamente impossibile o condurrebbe ad uno stato inconsistente. Una volta avviato il servizio non può fare altro che procedere fino al completamento.

Modalità di richiesta: segnale asserito per un singolo periodo di clock.

Es: Eliminazione di un elemento da una struttura dati.

INTERROMPIBILE: L'esecuzione del servizio può essere interrotta. Eventualmente l'interruzione potrebbe richiedere un ulteriore intervento da parte del richiedente per riportare il sistema in uno stato ben definito.

Modalità di richiesta: segnale asserito fino al completamento del servizio – oppure- segnale di interruzione dedicato.

Es: Ricerca all'interno di una struttura dati. Raggiungimento di un piano da parte di un ascensore.

REVOCABILE: (Tipico dei servizi duali) L'esecuzione del servizio può essere revocata, causando il ripristino dello stato precedente.

Modalità di richiesta: segnale asserito fino al completamento del servizio – oppure – deasserimento ed asserimento del segnale duale. Es: apertura del carrello del lettore CD. Porte di un ascensore.



Modellazione di servizi: durata

COMBINATORIA:

Il servizio è in realtà una funzione combinatoria di uno o più ingressi, che non altera lo stato del sistema, che viene risolto in un periodo di clock.

Modalità di sincronizzazione: nessuna, il richiedente sa che potrà leggere i risultati al clock successivo

SEQUENZIALE, NOTA A PRIORI:

Il servizio dura un numero prefissato di clock.

Modalità di interazione: nessuna, il richiedente sa quanto aspettare.

SEQUENZIALE, VARIABLE:

L'esecuzione del servizio può durare un numero variabile di cicli di clock.

Modalità di interazione: segnale di ACK per notificare completamento. (oppure segnale READY, vedi slide successive)



Modellazione di servizi: esito

IMPLICITO:

Non è necessario conoscere l'esito del servizio in quanto esso non può fallire (o non è possibile percepire un eventuale fallimento/malfunzionamento).

Modalità di notifica: nessuna.

Es.: notifica di un valore su un display. Trasmissione di uno stream di dati seriali su SPI.

SUCCESSO/FALLIMENTO:

IL servizio può fallire o comunque non essere in grado di completare.

Modalità di notifica: ACK + SUCCESS (il servizio è stato completato con/senza successo).

Es: Ricerca all'interno di una struttura dati. Raggiungimento di un piano da parte di un ascensore.

RISULTATO ESPPLICITO:

Il servizio prevede un risultato che va comunicato al richiedente

Modalità di notifica: ACK + risultato (+ SUCCESS)

Es: ricerca all'interno in una struttura dati.



Modellazione di servizi: concorrenza

ESCLUSIVO (UNIT-WIDE):

I vari servizi possono essere richiesti in maniera esclusiva all'unità. Eventualmente i segnali di ACK possono essere sostituiti / accompagnati da un unico segnale READY che determina se l'unità è in grado di accettare un nuovo servizio (o se ne sta già eseguendo uno)

SIMULTANEO (CON ALTRI SERVIZI):

Servizi distinti possono essere eseguiti in parallelo dall'unità.

PIPELINED:

Più istanze di uno stesso servizio (di durata NON unitaria) possono essere richieste in sequenza anche senza attendere il completamento delle precedenti. In realtà richiede molta attenzione e meriterebbe approfondimenti (che non abbiamo il tempo di effettuare).

In poche parole, l'unità deve comunicare (in modo distinto) quando è pronta per accettare un nuovo servizio (sebbene i precedenti non siano ancora completati) e deve comunicare quando un servizio (e quale!) “outstanding” è stato completato ed i risultati sono stati prodotti.

Es: è il tipico funzionamento delle ram dinamiche (SDRAM).



Obiettivi del Controller

I/O Giocatore

- BTN. LEFT
- BTN. RIGHT
- BTN. ACCEL
- BTN. ROTATE

1. Come gestiamo la discesa del *falling_piece*?
2. Come gestiamo il movimento orizzontale del *falling_piece*?
3. Quando e come verifichiamo il raggiungimento della posizione stabile del *falling_piece*? Come scegliamo il prossimo pezzo?
4. Quando e come controlliamo il completamento delle righe?
5. Cosa facciamo in caso di completamento di una riga?
6. Come interagiamo con il View?

Segnali datapath

CAN_MOVE_LEFT
CAN_MOVE_RIGHT
CAN_MOVE_DOWN
CAN_ROTATE
ROW_IS_COMPLETE

CLEAR
NEW_PIECE
MERGE
MOVE_DOWN
MOVE_LEFT
MOVE_RIGHT
ROTATE
ROW_INDEX[...]



Discesa del *falling_piece*

I/O Giocatore

-  BTN. LEFT
-  BTN. RIGHT
-  BTN. ACCEL
-  BTN. ROTATE

*Il falling piece cade “a tempo”, ma la sua discesa può essere accelerata dal giocatore.
(Inoltre in futuro la velocità di discesa potrebbe variare durante il gioco)*



Introduciamo un ingresso CLOCK_10MS come riferimento temporale. Lo gestiremo successivamente con una rete dedicata di distribuzione temporale.

Ruolo: Il segnale è asserito per un singolo periodo di CLOCK ogni 10 ms.

Segnali datapath

CAN_MOVE_LEFT
CAN_MOVE_RIGHT
CAN_MOVE_DOWN
CAN_ROTATE
ROW_IS_COMPLETE

CLEAR

NEW_PIECE

MERGE

MOVE_DOWN

MOVE_LEFT

MOVE_RIGHT

ROTATE

ROW_INDEX[...]

move_piece_down
std_logic

Introduciamo un segnale, asserito per un singolo periodo di clock, che indica la necessità di spostare il falling_piece giù di una posizione.

When to do

fall_speed
integer

Introduciamo un segnale che determina la velocità di discesa in riferimento alla base CLOCK_10MS.

Es fall_speed=3 -> il pezzo scende di una cella ogni 30 ms.

How to do

time_to_next_fall
integer

Introduciamo un contatore (all'indietro), che avanza con base di tempi CLOCK_10MS. Ogni qual volta il contatore raggiunge il valore 0, il pezzo viene spostato in basso (!) ed il contatore ricaricato.



VHDL per discesa *falling_piece*

```
architecture RTL of Tetris_Controller is
```

```
constant NORMAL_FALL_SPEED : integer := 50;  
constant FAST_FALL_SPEED   : integer := 10;
```

```
signal fall_speed           : integer range 1 to 100;  
signal time_to_next_fall   : integer range 0 to (fall_speed'high - 1);  
signal move_piece_down     : std_logic;
```

```
begin
```

```
fall_speed <= FAST_FALL_SPEED when (BUTTON_DOWN = '1') else STANDARD_FALL_SPEED;
```

```
TimedFall : process(CLOCK, RESET_N)
```

```
begin
```

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

```
time_to_next_fall <= 0;
```

```
move_piece_down   <= '0';
```

```
elsif rising_edge(CLOCK) then
```

```
move_piece_down <= '0';
```

```
if (TIME_10MS = '1') then
```

```
if (time_to_next_fall = 0) then
```

```
time_to_next_fall <= fall_speed - 1;
```

```
move_piece_down <= '1';
```

```
else
```

```
time_to_next_fall <= time_to_next_fall - 1;
```

```
end if;
```

```
end if;
```

```
end if;
```

```
end process;
```

I segnali assegnati sotto template
sincrono diventano registri

Move_piece_down di default a '0'.
(Monoimpulsore)

Assertito per un singolo ciclo di
CLOCK solo al wrap del contatore

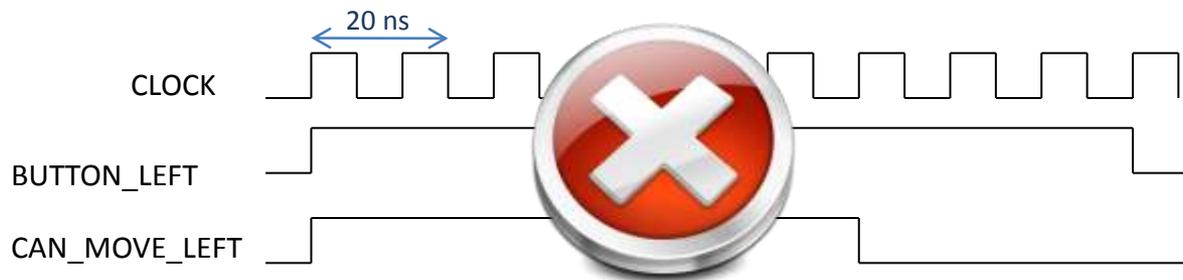
Movimento oriz. del *falling_piece*

I/O Giocatore

-  BTN. LEFT
-  BTN. RIGHT
-  BTN. ACCEL
-  BTN. ROTATE

Come generiamo il segnale MOVE_LEFT (ed analoghi)?

$MOVE_LEFT \leq BTN_LEFT \text{ and } CAN_MOVE_LEFT ?$

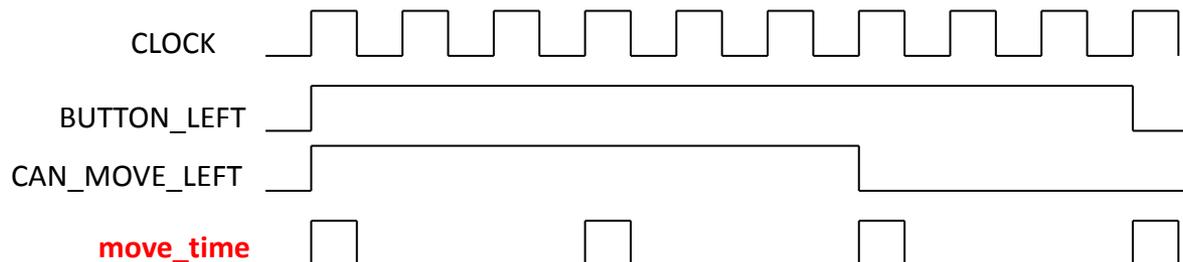


Per quanto possiamo essere veloci, la pressione del pulsante durerà molti cicli di clock. Di fatto il pezzo sarebbe incontrollabile e finirebbe sempre nelle posizioni più estreme.

Segnali datapath

- CAN_MOVE_LEFT
- CAN_MOVE_RIGHT
- CAN_MOVE_DOWN
- CAN_ROTATE
- ROW_IS_COMPLETE
- CLEAR
- NEW_PIECE
- MERGE
- MOVE_DOWN
- MOVE_LEFT
- MOVE_RIGHT
- ROTATE
- ROW_INDEX[...]

Analogamente alla discesa, sfruttiamo la base dei tempi





VHDL per abilitazione movimento *falling_piece*

architecture RTL of Tetris_Controller is

```
...  
constant MOVEMENT_SPEED      : integer := 10;  
...  
signal  time_to_next_move     : integer range 0 to MOVEMENT_SPEED-1;  
signal  move_time            : std_logic;
```

begin

TimedMovement : process(CLOCK, RESET_N)

begin

if (RESET_N = '0') then

time_to_next_move <= 0;

move_time <= '0';

elsif rising_edge(CLOCK) then

move_time <= '0';

if (TIME_10MS = '1') then

if (time_to_next_move = 0) then

time_to_next_move <= MOVEMENT_SPEED - 1;

move_time <= '1';

else

time_to_next_move <= time_to_next_move - 1;

end if;

end if;

end if;

end process;



VHDL Controller

```
signal random_piece : piece_type;
```

```
begin
```

```
Controller_RTL : process (CLOCK, RESET_N)
```

```
begin
```

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

```
--omissis (vedi sotto)
```

```
elsif rising_edge(CLOCK) then
```

```
MERGE <= '0';
```

```
NEW_PIECE <= '0';
```

```
MOVE_DOWN <= '0';
```

```
MOVE_LEFT <= '0';
```

```
MOVE_RIGHT <= '0';
```

```
ROTATE <= '0';
```

```
REDRAW <= '0';
```

```
row_check_req <= '0';
```

```
if (move_piece_down = '1') then
```

```
if (CAN_MOVE_DOWN = '1') then
```

```
MOVE_DOWN <= '1';
```

```
REDRAW <= '1';
```

```
else
```

```
MERGE <= '1';
```

```
NEW_PIECE <= '1';
```

```
NEW_PIECE_TYPE <= random_piece;
```

```
row_check_req <= '1';
```

```
end if;
```

```
elsif (move_time = '1') then
```

```
if (BUTTON_ROTATE = '1' and CAN_ROTATE = '1') then
```

```
ROTATE <= '1';
```

```
REDRAW <= '1';
```

```
elsif (BUTTON_LEFT='1' and CAN_MOVE_LEFT='1') then
```

```
MOVE_LEFT <= '1';
```

```
REDRAW <= '1';
```

```
elsif (BUTTON_RIGHT='1' and CAN_MOVE_RIGHT='1') then
```

```
MOVE_RIGHT <= '1';
```

```
REDRAW <= '1';
```

```
end if;
```

```
end if;
```

```
if (row_check_ack = '1') then
```

```
REDRAW <= '1';
```

```
end if;
```

```
end if;
```

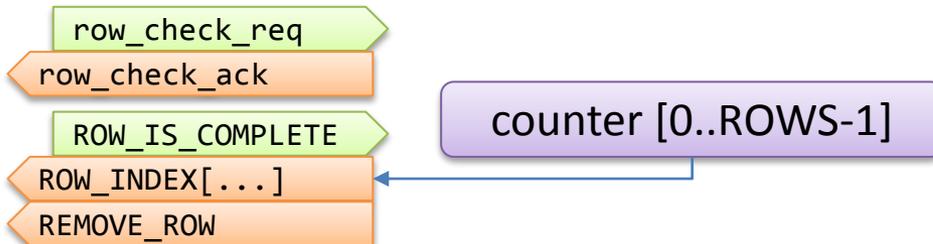
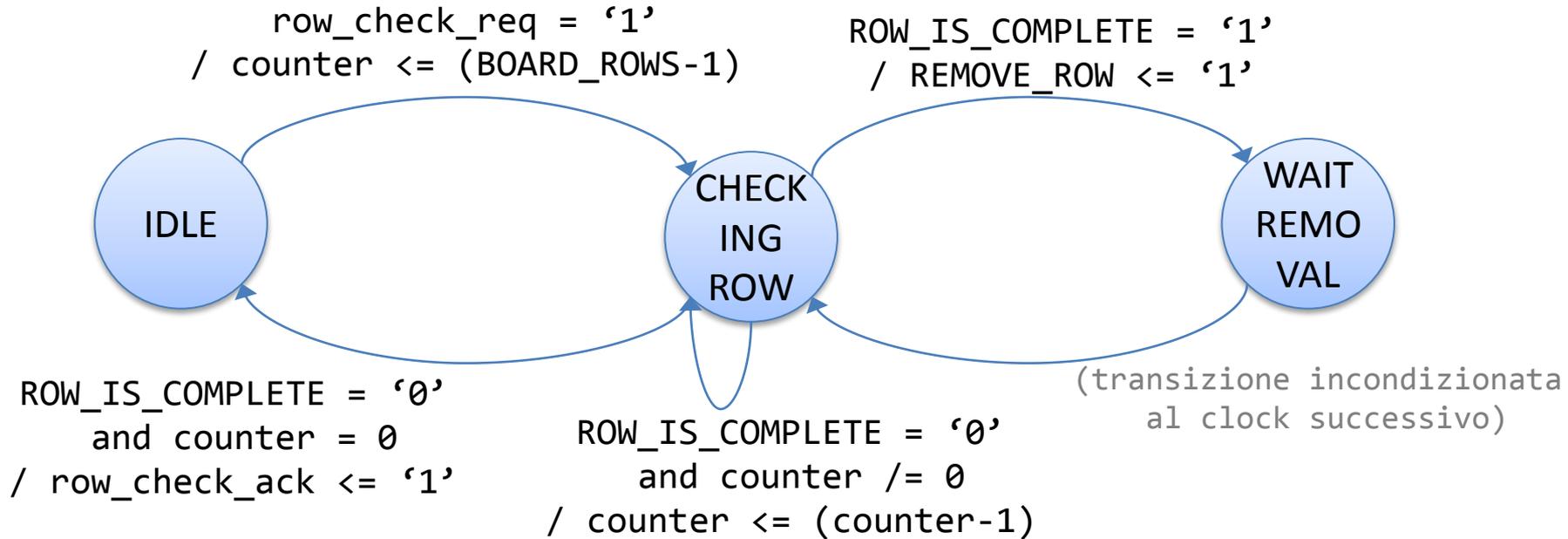
```
end process;
```

Generazione di random_piece...

a voi!



Controllo righe

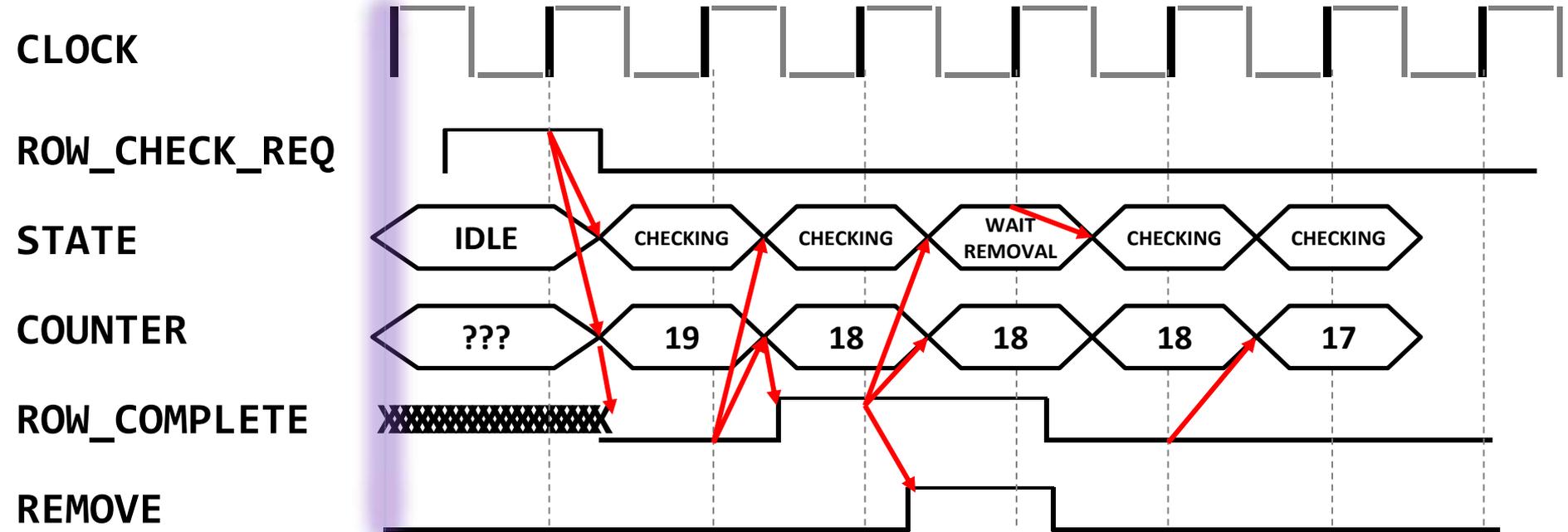


Nota:

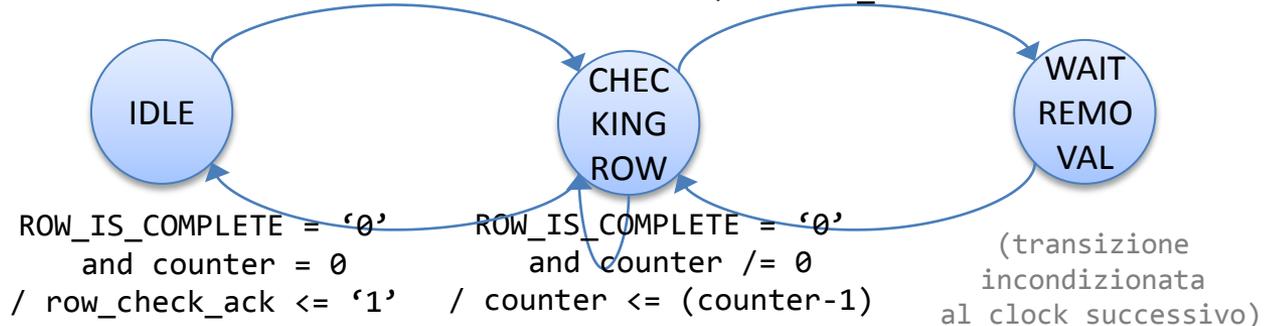
- tutte le uscite sono registrate, di default a '0', tranne quando assegnate (non ritentive)
- Tutte le transizioni avvengono solo sul fronte del clock



Controllo righe



row_check_req = '1'
 / counter <= (BOARD_ROWS-1) ROW_IS_COMPLETE = '1'
 / REMOVE_ROW <= '1'





VHDL per controllo righe

```
type row_check_state_type is
    (IDLE, CHECKING_ROW, WAIT_ROW_CHECK);
signal row_check_req      : std_logic;
signal row_check_ack      : std_logic;
signal row_check_counter : integer range 0 to BOARD_ROWS-1;
signal row_check_state    : row_check_state_type;

begin --architecture

Row_check : process(CLOCK, RESET_N)
begin

    if (RESET_N = '0') then
        REMOVE_ROW          <= '0';
        row_check_state     <= IDLE;
        row_check_ack       <= '0';

    elsif rising_edge(CLOCK) then
        REMOVE_ROW          <= '0';
        row_check_ack       <= '0';

        case (row_check_state) is

            when IDLE =>
                if (row_check_req = '1') then
                    row_check_state <= CHECKING_ROW;
                    row_check_counter <= BOARD_ROWS - 1;
                end if;

            when CHECKING_ROW =>
                if (ROW_IS_COMPLETE = '1') then
                    REMOVE_ROW          <= '1';
                    row_check_state     <= WAIT_ROW_REMOVAL;
                elsif (row_check_counter /= 0) then
                    row_check_counter <= row_check_counter-1;
                else
                    row_check_state     <= IDLE;
                    row_check_ack       <= '1';
                end if;

            when WAIT_ROW_REMOVAL =>
                row_check_state     <= CHECKING_ROW;

        end case;
    end if;
end process;
```