



VHDL

FSM - EXE

Testo

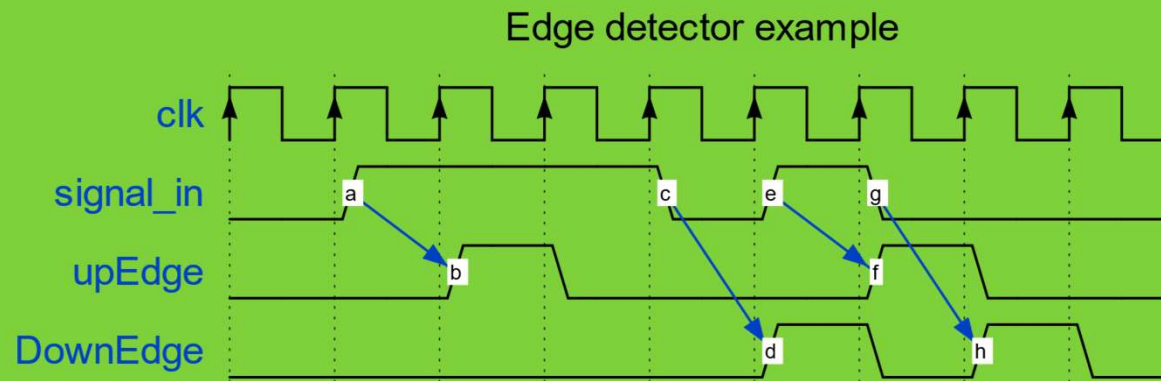
Creare un rilevatore di fronti positivi e negativi tramite una macchina a stati finiti di tipo Moore. Prima di iniziare con la scrittura di codice VHDL impostare uno schema a grafi identificando tutti gli stati e le varie condizioni particolari. Considerare il segnale in ingresso sincrono al clock che pilota tutta la macchina.

```
entity EdgeDetector is
  Port (
    clk      : in std_logic;
    reset    : in std_logic;

    signal_in : in std_logic;
    upEdge    : out std_logic;
    downEdge  : out std_logic;
  );
end EdgeDetector;
```

Testo

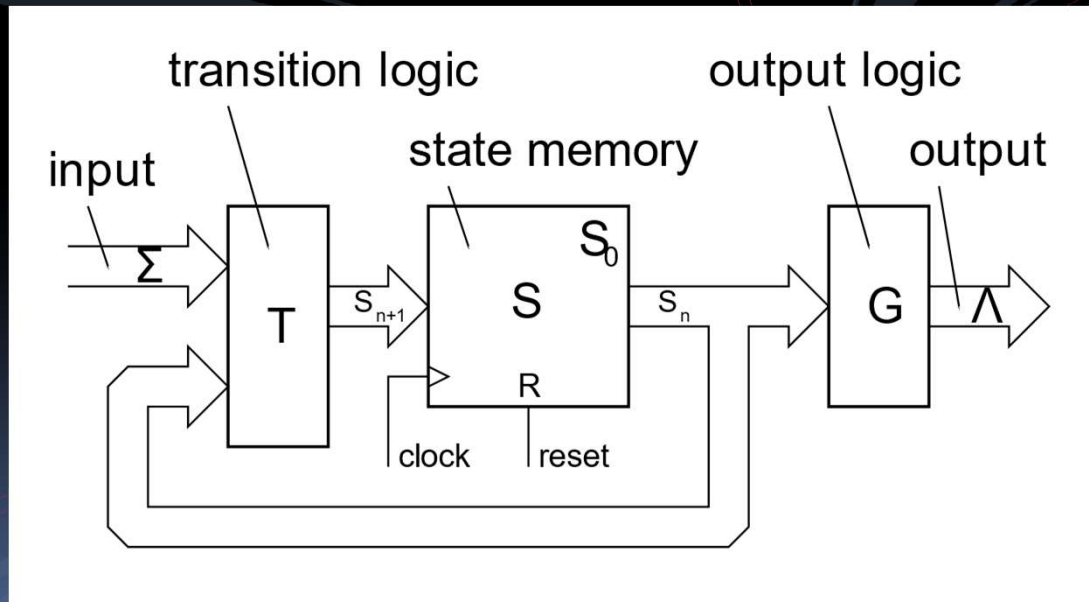
Qui un esempio di forme d'onda di un rilevatore di fronti:



Testo

Costruire la «FSM» come una macchina di Moore separando le tre parti fondamentali:

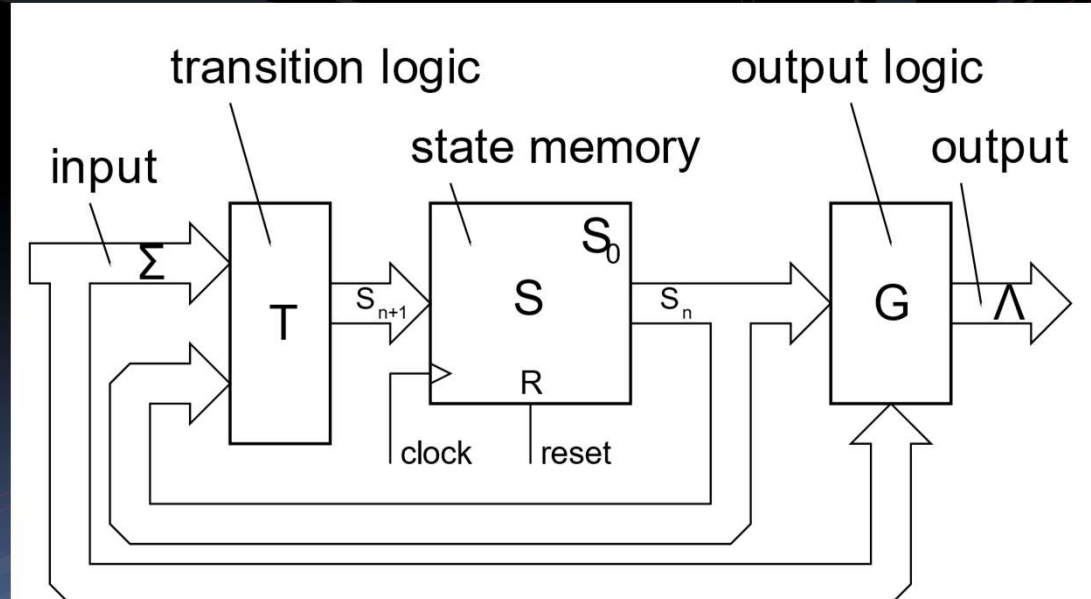
- Transition Logic
- State Register Synchronous Logic
- Output Logic



Testo Esercizio 2

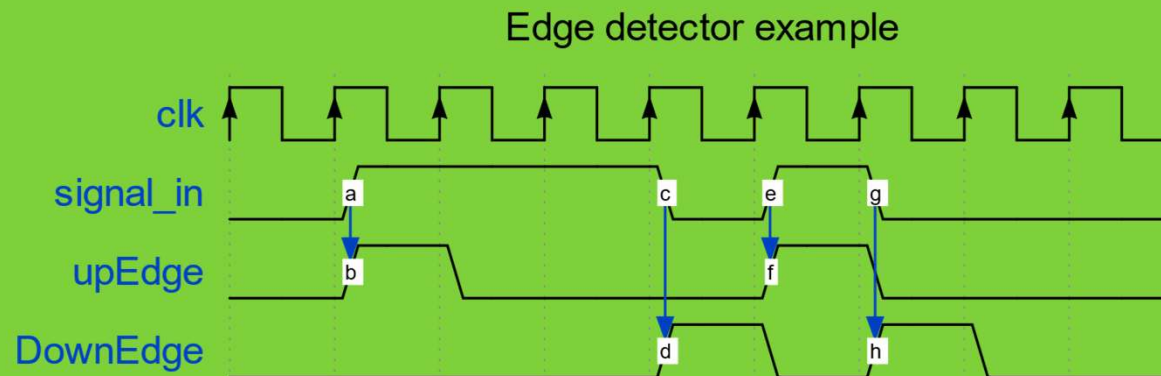
Costruire la «FSM» come una macchina di Mealy separando le tre parti fondamentali:

- Transition Logic
- State Register Synchronous Logic
- Output Logic



Testo Esercizio 2

Qui un esempio di forme d'onda di un rilevatore di fronti, notare la differenza con la macchina di Moore:





Tips

- Per separare le parti principali utilizzate tre «**process**» diversi
- Utilizzate due segnali di stato: **state** (lo stato attuale) e **nextState** (il prossimo stato)
- Come iniziare:
 - Disegnate il diagramma delle transizioni
 - Scrivete il codice VHDL seguendo il diagramma e la struttura delle FSM Moore/Mealy (slides 4/5)



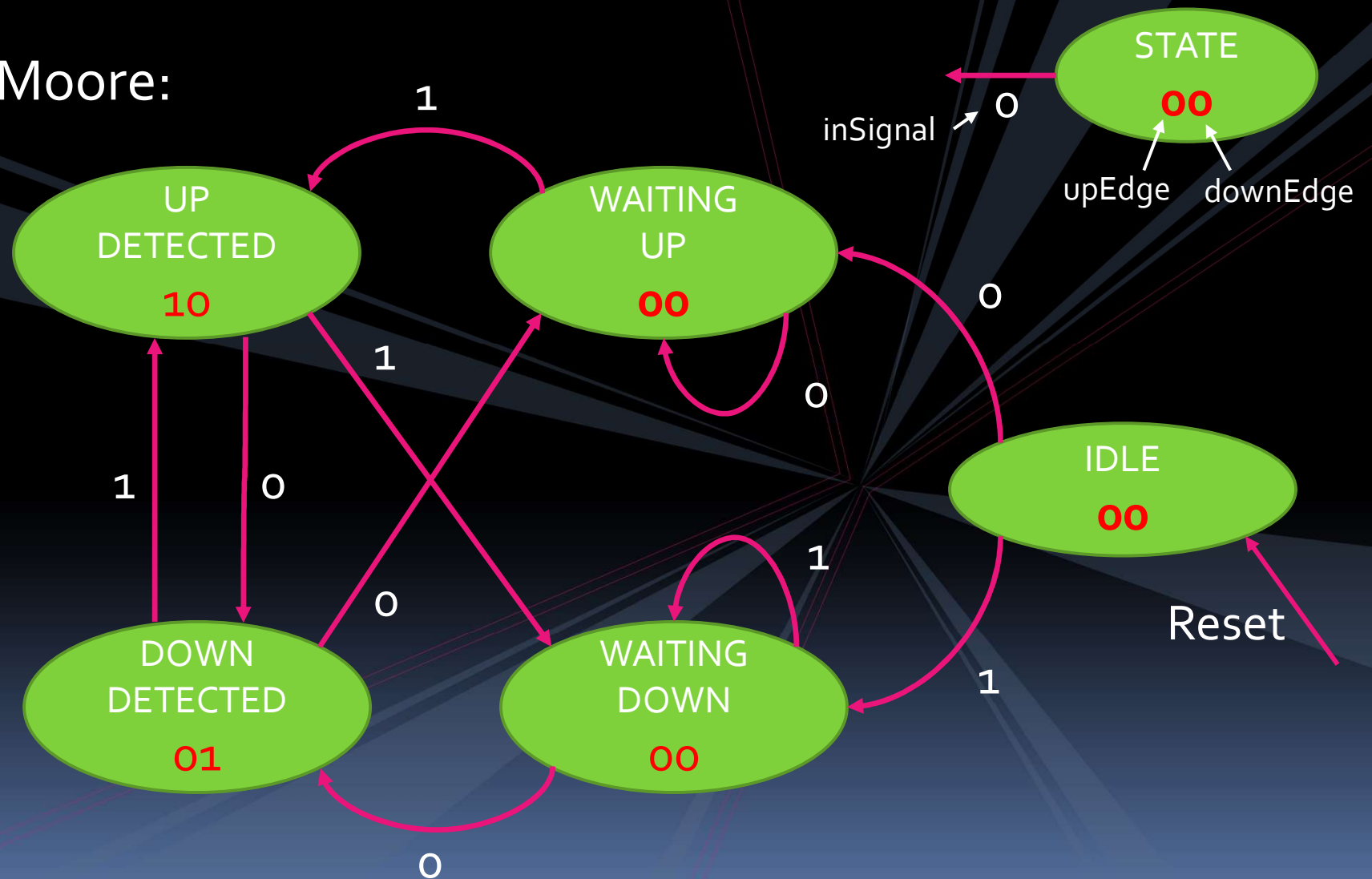
Consigli

Attenzione: di seguito alcune linee guida per arrivare ad una soluzione dell'esercizio.

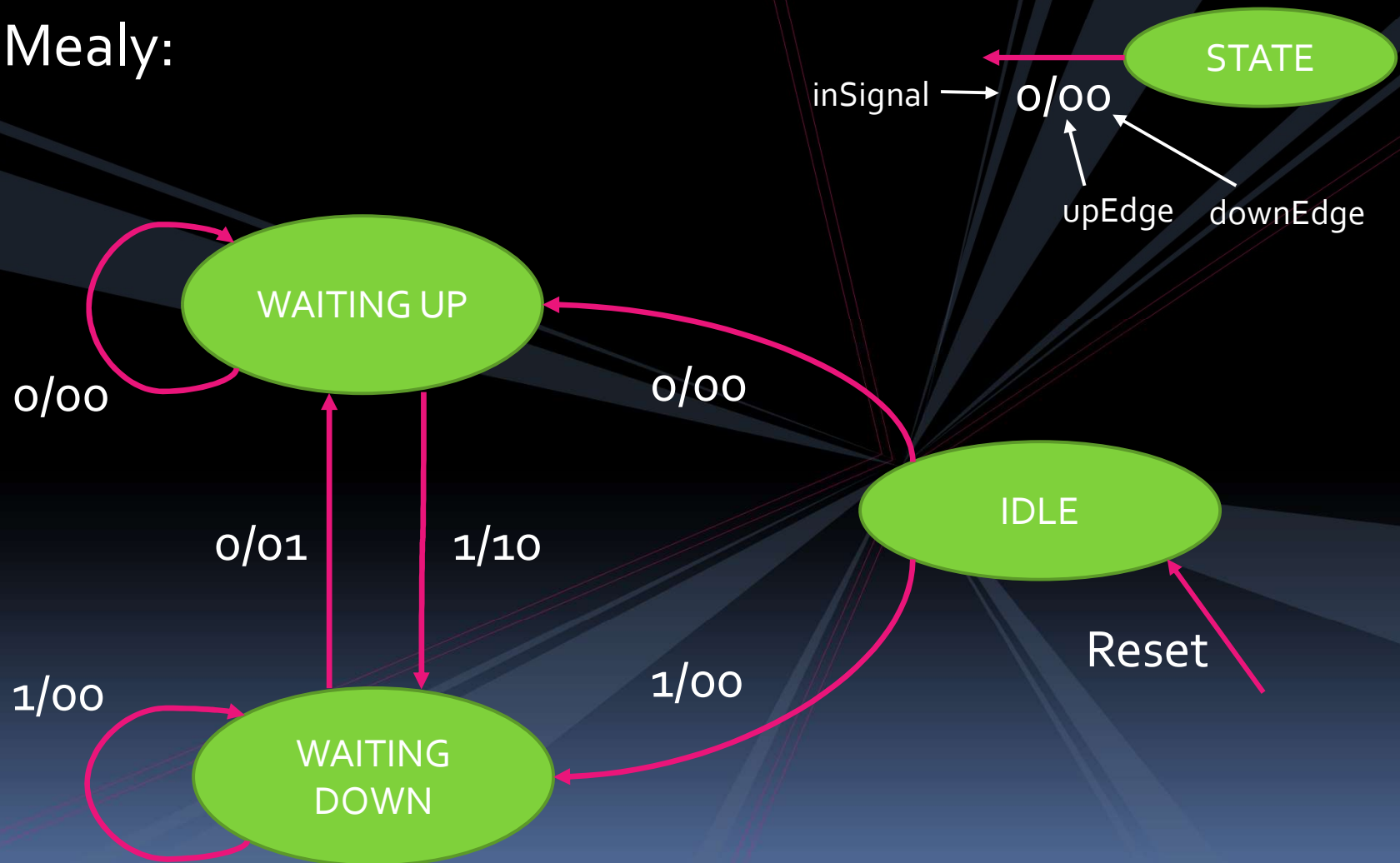
Consiglio di non leggerli prima di aver pensato autonomamente ad una soluzione.

Consigli

Moore:



Mealy:



Consigli

Utilizzate tre processi:

- **nextStateLogic**, per selezionare il prossimo stato
 - **ASINCRONO**
- **synchronousLogic**, per passare dallo stato attuale al successivo
 - **SINCRONO**
- **outputLogic**, per controllare gli output
 - **ASINCRONO**

```
nextStateLogic : process (...)  
begin  
  
end process;  
  
synchronousLogic : process (reset, clk)  
begin  
  
end process;  
  
outputLogic : process (...)  
begin  
  
end process;
```