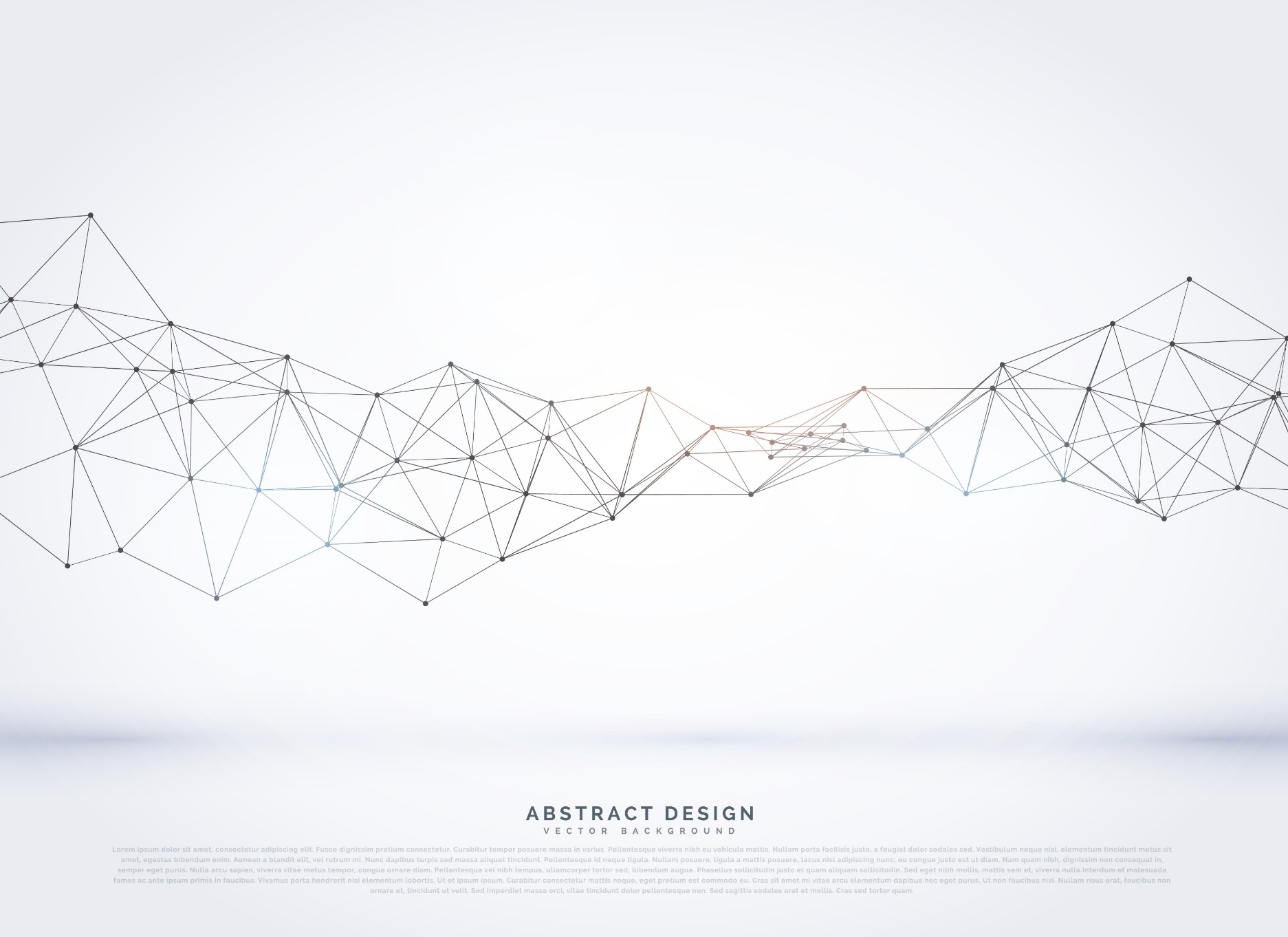


Corso di **Architettura dei Sistemi Digitali** – Prof. **Mazzocca**

Anno Accademico 2021-2022

Work Projects (Altro nome)

Nostri nomi

INDICE

[**1 Multiplexer** 5](#_Toc86833200)

[**1.1 Progettazione del Multiplexer** 5](#_Toc86833201)

[**1.1.2 Codice VHDL del Multiplexer** 5](#_Toc86833202)

[**1.1.3 La simulazione del Multiplexer** 5](#_Toc86833203)

[**2 Encoder BCD** 6](#_Toc86833204)

[**2.1 Progettazione del Encoder BCD** 6](#_Toc86833205)

[**2.1.1 Schematico del Encoder BCD** 6](#_Toc86833206)

[**2.1.2 Codice VHDL del Encoder BCD** 6](#_Toc86833207)

[**2.1.3 La simulazione del Encoder BCD** 6](#_Toc86833208)

[**3 Riconoscitore a 2 *modalità*** 7](#_Toc86833209)

[**3.1 Progettazione del Riconoscitore di sequenza a 2 modalità** 7](#_Toc86833210)

[**3.1.1 Schematico del Riconoscitore di sequenza a 2 modalità** 9](#_Toc86833211)

[**3.1.2 Codice VHDL del Riconoscitore di sequenza a 2 modalità** 10](#_Toc86833212)

[**3.1.3 La simulazione del Riconoscitore di sequenza a 2 modalità** 15](#_Toc86833213)

# **1 Multiplexer**

Si realizza un riconoscitore di sequenza a 2 modalità seguendo le specifiche richieste dal cliente:

* **Traccia**

## **1.1 Progettazione del Multiplexer**

### **1.1.2 Codice VHDL del Multiplexer**

### **1.1.3 La simulazione del Multiplexer**

# **2 Encoder BCD**

Si realizza un riconoscitore di sequenza a 2 modalità seguendo le specifiche richieste dal cliente:

* **Traccia**

## **2.1 Progettazione del Encoder BCD**

### **2.1.1 Schematico del Encoder BCD**

### **2.1.2 Codice VHDL del Encoder BCD**

### **2.1.3 La simulazione del Encoder BCD**

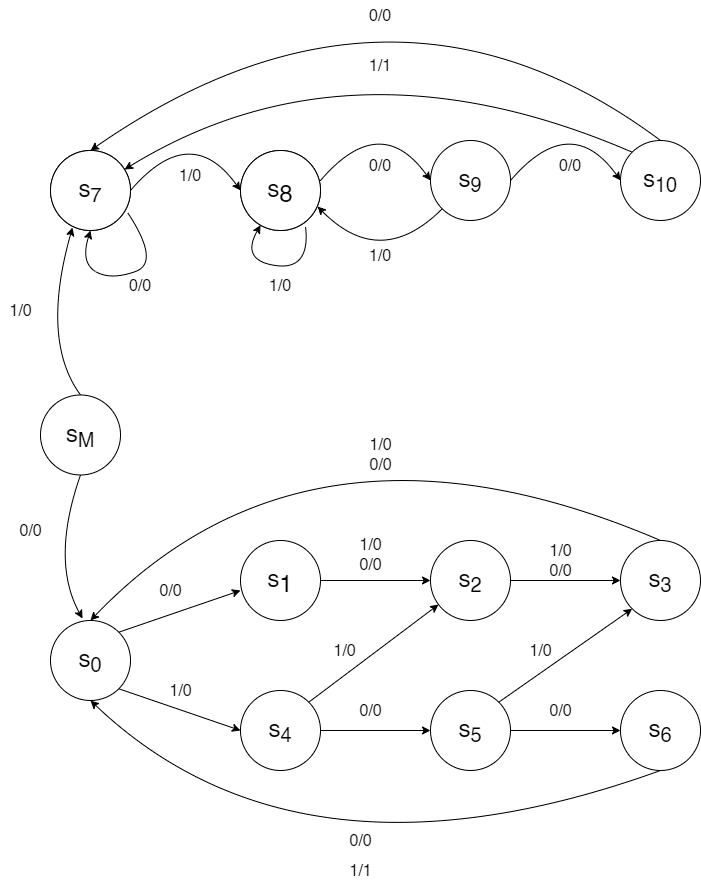
# **3 Riconoscitore a 2 *modalità***

Si realizza un riconoscitore di sequenza a 2 modalità seguendo le specifiche richieste dal cliente:

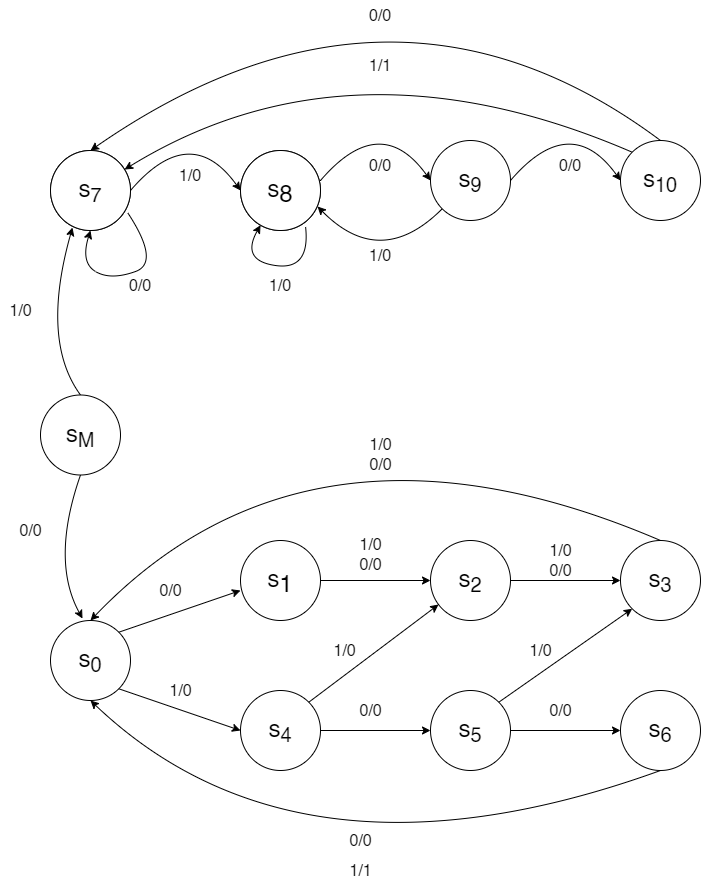
* **1.1** - Progettare, implementare in VHDL e testare mediante **simulazione** una macchina in grado di riconoscere la sequenza **1001**. La macchina prende in ingresso un segnale binario i che rappresenta il dato, un segnale A di tempificazione e un segnale M di modo, che ne disciplina il funzionamento, e fornisce un’uscita Y alta quando la sequenza viene riconosciuta. In particolare,
* se **M=0**, la macchina valuta i bit seriali in ingresso a gruppi di 4,
* se **M=1**, la macchina valuta i bit seriali in ingresso uno alla volta, tornando allo stato iniziale ogni volta che la sequenza viene correttamente riconosciuta.
* **1.2** - Sintetizzare e implementare su board la rete sviluppata al punto precedente, utilizzando uno switch S1 per codificare l’input i e uno switch S2 per codificare il modo M, in combinazione con due bottoni B1 e B2 utilizzati rispettivamente per acquisire l’input da S1 e S2 in sincronismo con il segnale di tempificazione A, che deve essere ottenuto a partire dal clock del board. Infine, l’uscita Y può essere codificata utilizzando un led.

## **3.1 Progettazione del Riconoscitore di sequenza a 2 modalità**

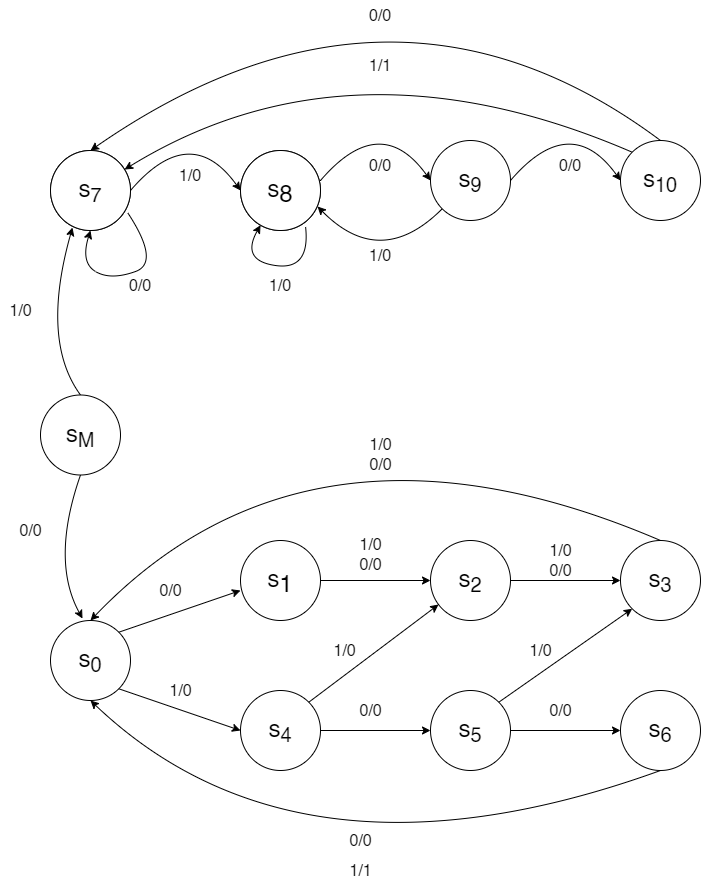
Per la realizzazione della macchina, si utilizza **un approccio comportamentale** dove viene descritto il comportamento della macchina come definito dai relativi automi. Le specifiche del progetto richiedono la valutazione dei bit in “2 modi” determinato dal segnale M di modo fornendo un’uscita Y alta quando la sequenza viene riconosciuta.   
Dalle indicazioni delle specifiche, la sequenza da riconoscere è la **1001**, pertanto si costruiscono le due macchine sequenziali:



**FIG. 1** - Automa che valuta i bit seriali in ingresso uno alla volta



**FIG. 2** - Automa che valuta i bit seriali in ingresso a gruppi di 4 bit.



**FIG. 3** - Automa con modo

Dalla **FIG.3** si evince dunque il funzionamento del riconoscitore. Partendo dallo stato iniziale **SM**, lo stato che determina il modo dell’automa, se M = 1, si valuta la sequenza **un** bit alla volta, se M = 0 si valuta la sequenza **a gruppi di 4 bit.**

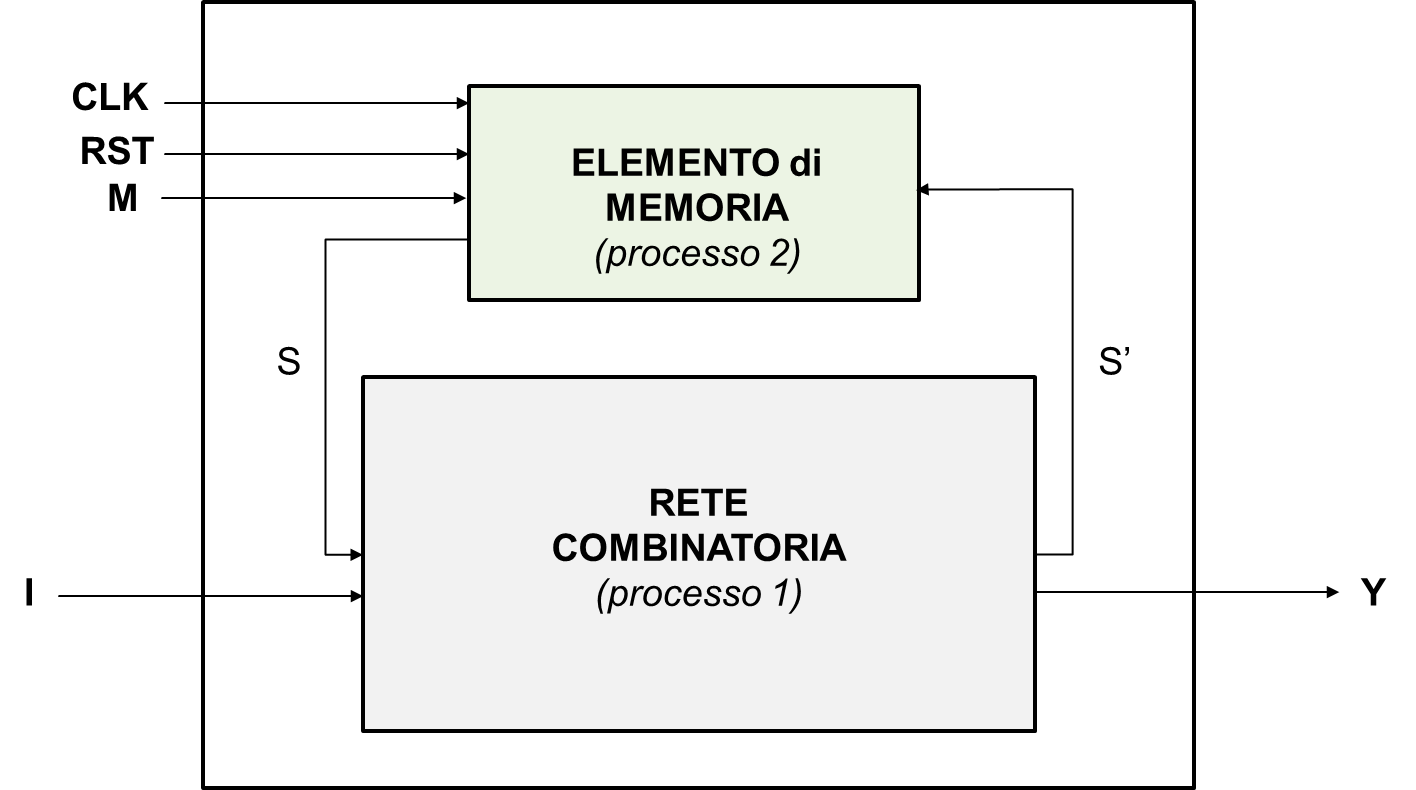
Dalle **FIGURE** si nota inoltre che è stato utilizzato il modello di **Mealy** dove la variazione dell’uscita a fronte di ingresso e di stato è:

(SERVE SCRIVERE ALTRO?)

### **3.1.1 Schematico del Riconoscitore di sequenza a 2 modalità**

Il modello del seguente riconoscitore è diviso da una rete combinatoria (la parte del sistema che calcolerà le funzioni di transizione da uno stato all’altro) e l’elemento di memoria (la retroazione) utile per la gestione SINCRONA, tramite RST (Reset) di M (Modo) tempificato dal CLK (Clock).

(SERVE SCRIVERE ALTRO?)



**FIG. 4** – Schematico del riconoscitore di sequenza

### **3.1.2 Codice VHDL del Riconoscitore di sequenza a 2 modalità**

Dallo schematico in **FIGURA 4**, si estrapolano i “process” da implementare in VHDL, nel caso in esame si sono utilizzati dunque due processi.

Si realizzano dunque due implementazioni: Design Sources per l’implementazione dei processi e un Simulation Sources per il TESTBENCH.

Si definisce **l’entità** composto da 4 porte in ingresso (I, RST, M, CLK) e l’uscita (Y).

**-- L’entità**

entity **Riconoscitore\_Mealy\_2\_Modi\_01** is

port( i: in std\_logic;

RST, CLK: in std\_logic;

M: in std\_logic;

Y: out std\_logic

);

end **Riconoscitore\_Mealy\_2\_Modi\_01**;

Nell’**architecture** si enumerano i 10 diversi stati che compongono entrambi gli automi e si pone come stato iniziale, come già detto, lo stato SM.

-- architettura

architecture **Behavioral** of **Riconoscitore\_Mealy\_2\_Modi\_01** is

-- enumerazione stati

type **stato** is **(SM, S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10)**;

-- stato iniziale SM

signal **stato\_corrente** : stato := **SM**;

signal **stato\_prossimo** : stato;

Il processo **combinatorio** sarà così costituito:

begin

stato\_uscita: **process**(stato\_corrente, i)

begin

-- se M = 0 : Riconoscitore di Sequenza con valutazione di sequenza per gruppi di 4 bit

-- se M = 1 : Riconoscitore di Sequenza con valutazione di sequenza ad ogni singolo bit

case **stato\_corrente** is

when **S0** =>

if(i = '0' ) then

**stato\_prossimo** <= S1;

Y <= '0';

else

**stato\_prossimo** <= S4;

Y <= '0';

end if;

when **S1** =>

if( i = '0' or i = '1' ) then

**stato\_prossimo** <= S2;

Y <= '0';

end if;

when **S2** =>

if( i = '0' or i = '1' ) then

**stato\_prossimo** <= S3;

Y <= '0';

end if;

when **S3** =>

if( i = '0' or i = '1' ) then

**stato\_prossimo** <= S0;

Y <= '0';

end if;

when **S4** =>

if( i = '0' ) then

**stato\_prossimo** <= S5;

Y <= '0';

else

**stato\_prossimo** <= S2;

Y <= '0';

end if;

when **S5** =>

if( i = '0' ) then

**stato\_prossimo** <= S6;

Y <= '0';

else

**stato\_prossimo** <= S3;

Y <= '0';

end if;

when **S6** =>

if( i = '0' ) then

**stato\_prossimo** <= S0;

Y <= '0';

else

**stato\_prossimo** <= S0;

Y <= '1';

end if;

when **S7** =>

if( i = '0' ) then

**stato\_prossimo** <= S7;

Y <= '0';

else

**stato\_prossimo** <= S8;

Y <= '0';

end if;

when **S8** =>

if( i = '0' ) then

**stato\_prossimo** <= S9;

Y <= '0';

else

**stato\_prossimo** <= S8;

Y <= '0';

end if;

when **S9** =>

if( i = '0' ) then

**stato\_prossimo** <= S10;

Y <= '0';

else

**stato\_prossimo** <= S8;

Y <= '0';

end if;

when **S10** =>

if( i = '0' ) then

**stato\_prossimo** <= S7;

Y <= '0';

else

**stato\_prossimo** <= S7;

Y <= '1';

end if;

when **others** =>

**stato\_prossimo** <= S7;

Y <= '0';

end **case**;

end **process**;

Si è usato il costrutto **CASE-WHEN** per definire cosa fa la macchina sequenziale in ogni stato. Ogni valore dell’ingresso ‘I’ fa evolvere la macchina verso un nuovo stato specificando dunque una determinata uscita Y.

Il processo **di memorizzazione** sarà:

mem: **process** (CLK)

begin

if(CLK'**event** and CLK = '1') then

if(RST = '1') then

**stato\_corrente** <= SM;

if(M = '0') then

**stato\_corrente** <= S0;

else

**stato\_corrente** <= S7;

end if;

else

**stato\_corrente** <= **stato\_prossimo**;

end **if**;

end **if**;

end **process**;

end **Behavioral**;

L’ istruzione consente di cambiare la modalità del riconoscitore soltanto quando il **RST** è abilitato a **1**. Se M = 0, si imposta lo stato corrente del riconoscitore a 4 bit, se M = 1, si imposta lo stato corrente del riconoscitore a 1 bit.​

L’ assegnazione **stato\_corrente** <= **stato\_prossimo**; consente di retroazionare lo stato prossimo nello stato corrente​

(SERVE ALTRO?)

Il TB ci permette di simulare il funzionamento della macchina in particolare si può apprezzare il comportamento del Riconoscitore nei casi in cui M = 0 o 1 e come le abilitazioni del RST influiscano sul cambio di modalità della macchina.

entity **Riconoscitore\_Mealy\_2\_Modi\_01\_TB** is

-- Port ( );

end **Riconoscitore\_Mealy\_2\_Modi\_01\_TB**;

architecture Behavioral of **Riconoscitore\_Mealy\_2\_Modi\_01\_TB** is

COMPONENT **Riconoscitore\_Mealy\_2\_Modi\_01\_TB**

PORT(

i : IN std\_logic;

CLK,RST,M : IN std\_logic;

Y : OUT std\_logic

);

END **COMPONENT**;

-- Inputs

signal i : std\_logic := '0';

signal CLK : std\_logic := '0';

signal RST : std\_logic := '0';

signal M : std\_logic := '1';

--Outputs

signal Y : std\_logic;

-- Clock period definitions

constant CLK\_period : time := 10 ns;

begin

-- Instantiate the Unit Under Test (UUT)

-- Qui si specifica quale architecture simulare di quelle definite nel progetto corrente

uut: entity work.**Riconoscitore\_Mealy\_2\_Modi\_01**(Behavioral) port map(

i => i,

CLK => CLK,

RST => RST,

M => M,

Y => Y

);

-- Clock process definitions

CLK\_process :process

begin

CLK <= '0';

wait for CLK\_period/2;

CLK <= '1';

wait for CLK\_period/2;

end process;

-- Stimulus process

stim\_proc: process

begin

-- hold reset state for 100 ns.

wait for 100 ns;

-- start da 200 (100 + CLK\_period\*10)

wait for CLK\_period\*10;

-- insert stimulus here

-- sequenza di bit in ingresso 0010010000101001

i<='0';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='1';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='1';

wait for 10 ns;

i<='0';

wait for 10 ns;

--C'è il cambio di M e viene abilitato il reset (RST<='1')

M<='0';

RST<='1';

wait for 10 ns;

RST<='0';

i<='0';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='1';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='1';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='1';

wait for 10 ns;

-- C'è il cambio di M e viene abilitato il reset (RST<='1')

M<='1';

RST<='1';

wait for 10 ns;

RST<='0';

i<='0';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='1';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='1';

wait for 10 ns;

i<='0';

wait for 10 ns;

i<='0';

wait for 10 ns;

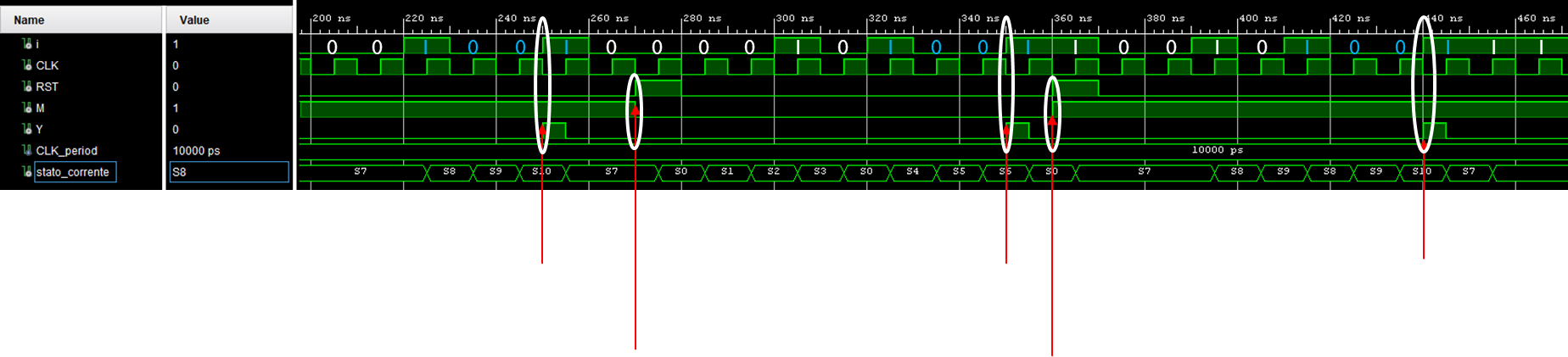
i<='1';

wait;

end **process**;

end **Behavioral**;

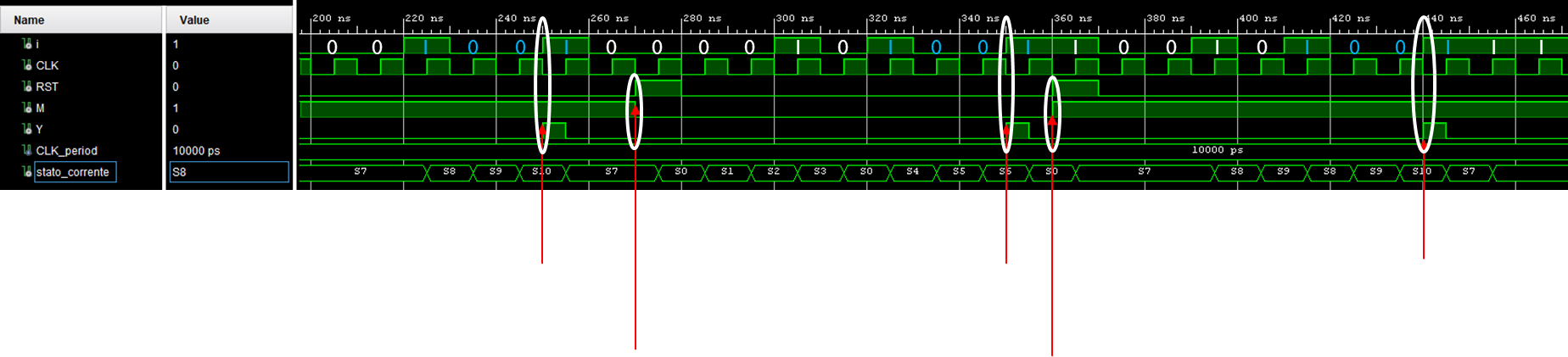
### **3.1.3 La simulazione del Riconoscitore di sequenza a 2 modalità**

Seguendo il TESTBENCH, la simulazione sarà:

A ***270ns*** si ha la transizione di M: 1 🡪 0 e RST: 0 🡪 1

(RST torna a 0 dopo 10ns)

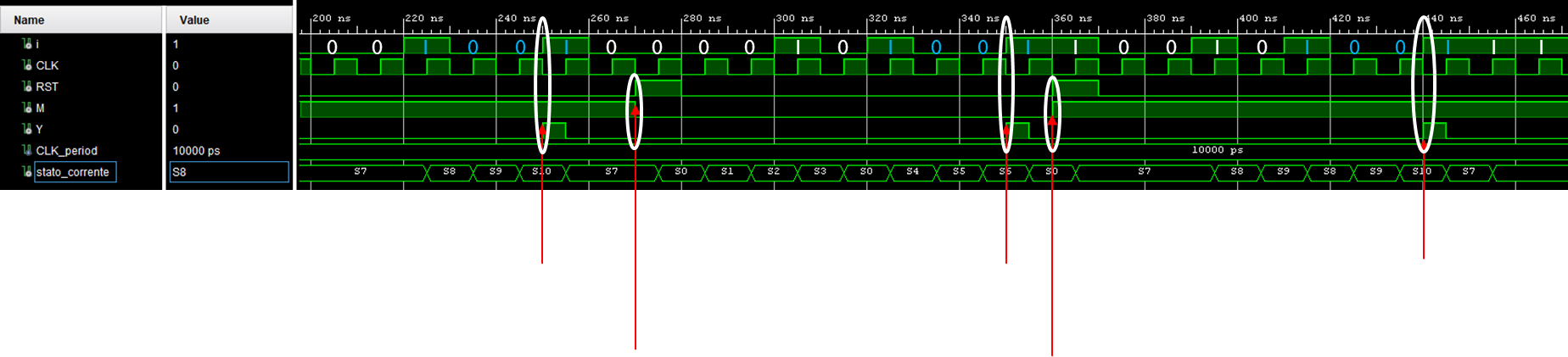
A ***250ns*** si ha la transizione di Y: 0 🡪 1 (M=1)



A ***350ns*** si ha la transizione di Y: 0 🡪 1 (M=0)

A ***360ns*** si ha la transizione di M: 0 🡪 1 e RST: 0 🡪 1

(RST torna a 0 dopo 10ns)



A ***440ns*** si ha la transizione di Y: 0 🡪 1 (M=1)

VEDERE LA QUESTIONE DEGLI ASSERT