

# RELAZIONE PROVA FINALE RETI LOGICHE A.A. 2022/2023

FIORINI ARMANDO  
(Cod. Persona 10709856)

March 2023

## 1 INTRODUZIONE

L'obiettivo della prova è la progettazione di un componente hardware che, interfacciandosi con una memoria, fornisca in output, su un'uscita da 8 bit specificata in input tra le 4 disponibili, il dato contenuto in memoria all'indirizzo anch'esso fornito in input (i\_w).

Il componente deve inoltre essere dotato di un segnale di reset, che può essere attivato ( $\text{reset} = 1$ ) in qualsiasi momento e lo riporta allo stato iniziale, e due segnali di start e done: il primo viene messo a 1 quando inizia una nuova sequenza di bit da leggere in input e indica perciò una nuova richiesta a memoria: il segnale di start rimane alto per almeno 2 cicli di clock e per non più di 18 cicli; nella lettura, che deve avvenire sempre durante il fronte di salita del clock, i primi due bit ricevuti codificano l'uscita a cui il dato dovrà essere inviato (00 è la prima uscita, 01 la seconda e così via) mentre gli altri (da 0 a 16) a cui vanno aggiunti gli opportuni 0 di estensione a sinistra, identificano un indirizzo di memoria a 16 bit da cui prendere il dato nella memoria; tutti i 18 bit sono forniti sempre a partire da quello più significativo.

Il secondo segnale (o\_done) assume il valore 1 per un solo ciclo di clock quando la richiesta è completata e il dato richiesto è disponibile e visualizzabile in corrispondenza dell'uscita specificata: in corrispondenza di ciò su ognuna delle altre tre uscite deve essere visualizzabile l'ultimo valore che quell'uscita ha visualizzato in seguito a precedenti letture da memoria, se ci sono state; al ciclo di clock successivo il segnale o\_done torna a 0 e le uscite tornano ad assumere tutte valore 0;

Il componente deve avere la seguente interfaccia:

```
entity project_reti_logiche is
Port (
    i_clk : in std_logic;
```

```

    i_rst : in std_logic;
    i_start : in std_logic;
    i_w : in std_logic;
    o_z0 : out std_logic_vector(7 downto 0);
    o_z1 : out std_logic_vector(7 downto 0);
    o_z2 : out std_logic_vector(7 downto 0);
    o_z3 : out std_logic_vector(7 downto 0);
    o_done : out std_logic;
    o_mem_addr : out std_logic_vector(15 downto 0);
    i_mem_data : in std_logic_vector(7 downto 0);
    o_mem_we : out std_logic;
    o_mem_en : out std_logic
);
end project_reti_logiche;

```

La memoria, il cui funzionamento è descritto all'interno dei test bench e non fa parte del componente, contiene dati da 8 bit, contenuti in celle identificate da indirizzi da 16 bit, e possiede 4 segnali che il componente deve controllare: uno di enable (o\_mem\_en), che permette al componente di entrare in comunicazione dalla memoria, uno di scrittura (o\_mem\_we), che se posto a 0 abilita la lettura da memoria mentre se a 1 la scrittura, un segnale o\_mem\_address, in cui viene inserito l'indirizzo di memoria da cui vogliamo leggere (o da cui vogliamo scrivere, cosa che il componente non fa), e il segnale i\_mem\_data, in cui la memoria carica il dato letto.

## 2 ARCHITETTURA

La realizzazione del progetto si basa sull'idea di una FSM completamente specificata a 8 stati, di cui 1 di reset, il cui funzionamento è specificato attraverso 2 processi: uno che rappresenta lo state register, che gestisce la transizione di stato della macchina, e uno che realizza contemporaneamente la funzione di uscita e quella di stato prossimo: il cambiamento di stato della macchina avviene sempre durante il fronte di discesa del clock, mentre le operazioni relative al secondo processo vengono effettuate sempre durante il fronte di salita.

La FSM specificata è così rappresentabile:



signal curr\_st,next\_st: S; - rappresentano rispettivamente lo stato corrente e lo stato prossimo della FSM;

## **2.2 Descrizione degli stati**

### **2.2.1 SR**

Stato di reset della macchina, porta a 0 tutti i valori delle uscite, dei segnali R1,...;R5, del segnale di done e di quello relativo all'indirizzo di memoria da cui leggere: si entra in questo stato quando il segnale i\_rst si alza a 1 e si esce quando ritorna a 0, entrando nello stato S0.

### **2.2.2 S0**

Stato iniziale della macchina appena avviata e prima di ogni lettura e raggiunto dalla macchina al termine di ogni richiesta, riporta il Done e le uscite al valore 0 insieme a R0, segnale che contiene l'indirizzo d inviare in input alla memoria: quando il segnale di start sale a 1 da S0 si raggiunge S1, e, associata a questa transizione, abbiamo la lettura del primo bit dei due che indicano l'uscita dove mostrare il dato che sarà letto dalla memoria, che viene inserito nella posizione più significativa del segnale apposito R1.

### **2.2.3 S1**

Lo stato S1 viene utilizzato per leggere il secondo bit di codifica dell'uscita, che viene memorizzato nella posizione meno significativa del segnale R1.

### **2.2.4 S2**

Lo stato S2 viene utilizzato per leggere, fintanto che il segnale i\_start rimane sul valore alto, un bit da i\_w, che viene memorizzato in R0 dopo uno shift verso sinistra: in questo modo il primo bit letto sarà sempre in posizione più significativa (fatta eccezione per gli 0 di estensione nel caso i bit letti fossero meno di 16), il secondo nella seconda e così via.

### **2.2.5 S3**

In questo stato l'indirizzo in R0 viene passato alla memoria e viene posto a 1 il segnale di enable, che, abbinato al segnale di write enable lasciato fisso a 0, consente di leggere il dato dalla memoria.

### **2.2.6 S4**

Stato in cui viene disabilitato il segnale di enable della memoria, il salvataggio del dato in uscita avviene in uno stato successivo per questione di sincronismo.





### 3.3 CONCLUSIONI

In conclusione il componente risulta soddisfare la specifica assegnata in maniera efficiente ed ottimizzata, sia dal punto di vista dell'analisi comportamentale, sia nell'analisi post-sintesi, eseguendo quanto richiesto in tempo anche molto breve rispetto alla soglia massima consentita, di fronte a test-bench che ne hanno garantito la sottoposizione a tutti i possibili casi limite ma anche a quelli più generici.

La realizzazione del progetto attraverso una macchina a stati ne ha consentito una facile scalabilità e in un approccio proiettato fin dal principio al livello hardware e alla sintesi logica, evitando che quest'ultima creasse particolari problemi e consentendo al componente di risultare anche correttamente implementabile (Ho testato la macchina anche in post-implementation pur non essendo richiesto e risulta superare i test).