POLITECNICO
MILANO 1863

Prova Finale di Reti Logiche
Anno Accademico 2023/2024

Scianna Marco
Codice Persona: 10815900 Matricola: 984631

Villa Alessio
Codice Persona: 10791818 Matricola: 982446

Indice

1 Introduzione

1.1 Scopo del Progetto e Specifiche

Il presente studio si focalizza sull'analisi e l'implementazione di un modulo progettato per gestire sequenze di parole all'interno di un sistema di elaborazione dati. Tale modulo è progettato per completare sequenze di parole, sostituendo i valori mancanti con quelli precedentemente letti e per valutare la credibilità dei dati tramite un'apposita metrica. L'obiettivo principale è garantire un'elaborazione accurata dei dati, soprattutto in presenza di informazioni incomplete.

Questo modulo è progettato per gestire sequenze di parole, in cui ogni parola è rappresentata da un valore compreso tra 0 e 255. La particolarità risiede nel fatto che il valore 0 viene interpretato come un'indicazione che "il valore non è specificato". Le parole sono memorizzate seguendo un preciso schema di indirizzamento, consentendo al modulo di individuare e completare eventuali lacune nei dati.

Il modulo si occupa del completamento delle sequenze, sostituendo i valori mancanti con l'ultimo valore non nullo letto precedentemente e inserendo un valore di "credibilità" per ogni parola della sequenza. La credibilità associata a ciascuna parola è valutata in base al valore W della sequenza. Quando il valore di W è diverso da zero, la credibilità è impostata a 31; viceversa, viene decrementata ogni volta che si incontra uno zero in W.

La sostituzione dei valori nulli avviene copiando l'ultimo valore valido precedentemente letto. La credibilità di ogni parola W viene memorizzata in memoria nel byte immediatamente successivo. È importante sottolineare che il valore di credibilità è sempre maggiore o uguale a 0 ed è reinizializzato a 31 ogni volta che si incontra un valore W diverso da zero. Una volta che il valore di credibilità raggiunge lo 0, non viene ulteriormente decrementato.

1.2 Esempio di Funzionamento

La figura 1 descrive in modo esemplificativo il funzionamento del modulo.

Esempio:

Sequenza di input: 128 0 64 0 0 0 0 0 0 0 0 0 0 0 100 0 1 0 0 0 5 0 23 0 200 0 0 0

Sequenza di output: 128 31 64 31 64 30 64 29 64 28 64 27 64 26 100 31 1 31 1 30 531 23 31 200 31 200 30

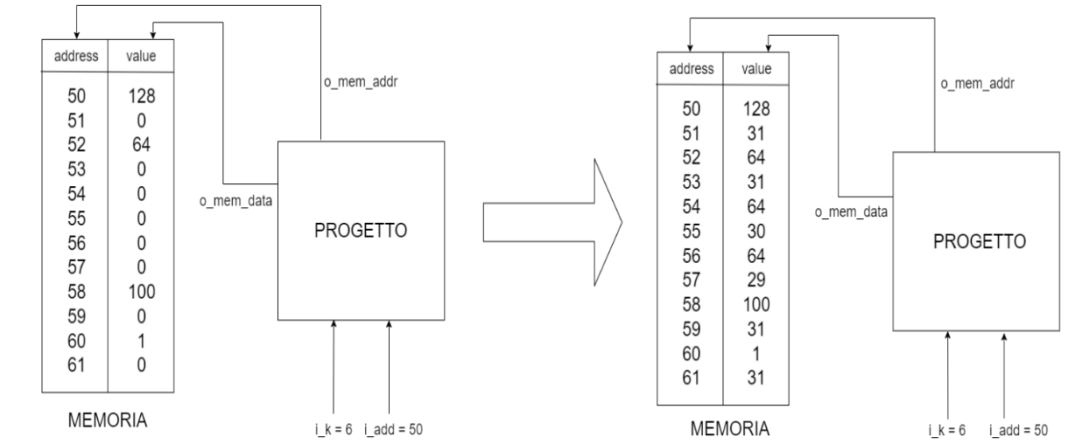


Figura 1

1.3 Interfaccia del Componente

Il componente è raffigurato da un punto di vista esterno in figura 2.

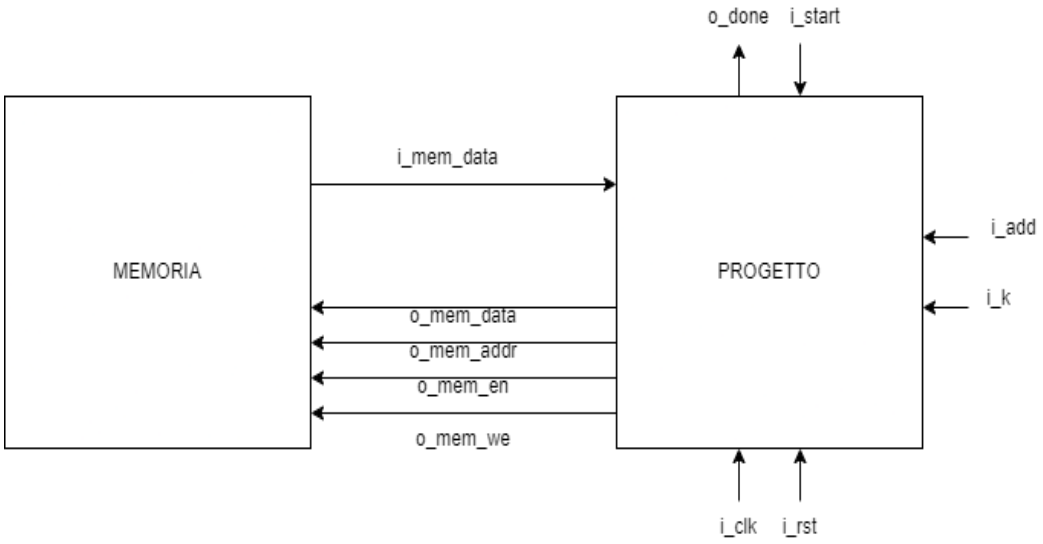


Figura 2

Oltre ai segnali forniti, ne abbiamo creati cinque ulteriori. Questa è la lista di tutti i seganli:
i_clk: in std_logic;
i_rst: in std_logic;
i_start: in std_logic;
i_add: in std_logic_vector(15 downto 0);

```
i_k : in  std_logic_vector(9 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_data : out std_logic_vector(7 downto 0);
o_mem_we: out std_logic;
o_mem_en: out std_logic;

signal state: state_type;
signal int_address: std_logic_vector(15 downto 0);
signal k: integer range 0 to 1023;
signal reg: std_logic_vector(7 downto 0);
signal C: std_logic_vector(7 downto 0);
```

2 Architettura

2.1 Scelte Progettuali

Inizialmente, si è progettata una serie di componenti al fine di implementare le funzionalità richieste. Questi componenti sono orchestrati da una macchina a stati finiti (FSM) che gestisce il flusso del processo attraverso i suoi stati e i segnali di enable diretti verso gli altri elementi del sistema. Inoltre, la FSM è responsabile delle operazioni di scrittura e lettura in memoria.
Essa presenta anche tre segnali aggiuntivi: zero_start, credibility_reset e credibility_decrease, i cui nomi sono autoesplicativi ma vengono comunque discussi in dettaglio nei componenti successivi:

- Counter: è dedicato alla memorizzazione e manipolazione del valore di credibilità. Esso interviene nel reset dell'intero sistema, reimpostando il valore di credibilità a 31 con il segnale credibility_reset. Inoltre, tramite il segnale credibility_decrease, il valore viene decrementato di 1, mentre con zero_start viene azzerato;
- Register: svolge il compito di conservare le parole W ricevute in ingresso da i_mem_data, eccezion fatta per il valore zero iniziale;
- Mux: è responsabile della selezione e dell'instradamento dei dati verso la memoria, inviando il valore o_mem_data da salvare;
- Memoria è il componente che effettivamente contiene i valori.

La rappresentazione dei componenti e dei segnali è fornita nella Figura 3.

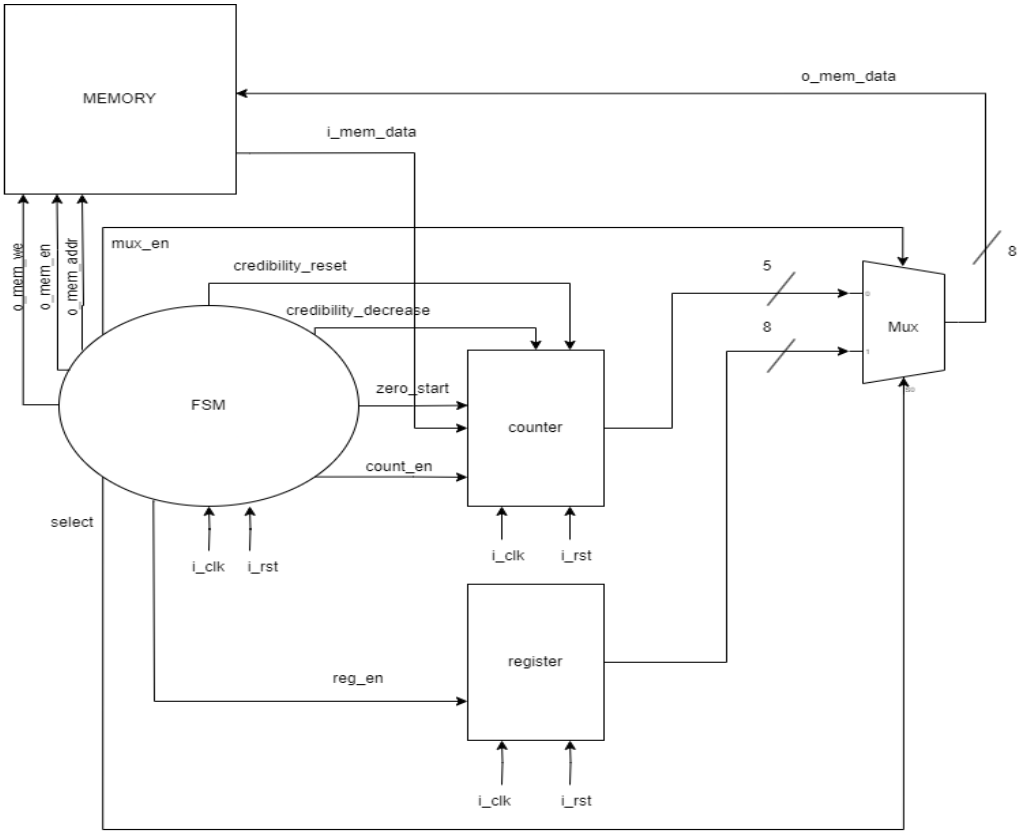


Figura 3

2.2 Soluzione Finale

Per garantire implementazioni più veloci e tempi di esecuzione ridotti, si è deciso di adottare un approccio basato su una singola FSM composta da 10 stati (figura 4). Tale scelta è stata motivata dalla sua capacità di gestire efficacemente le complesse operazioni richieste dal progetto, risultando pertanto più adatta alle dimensioni del lavoro in questione.

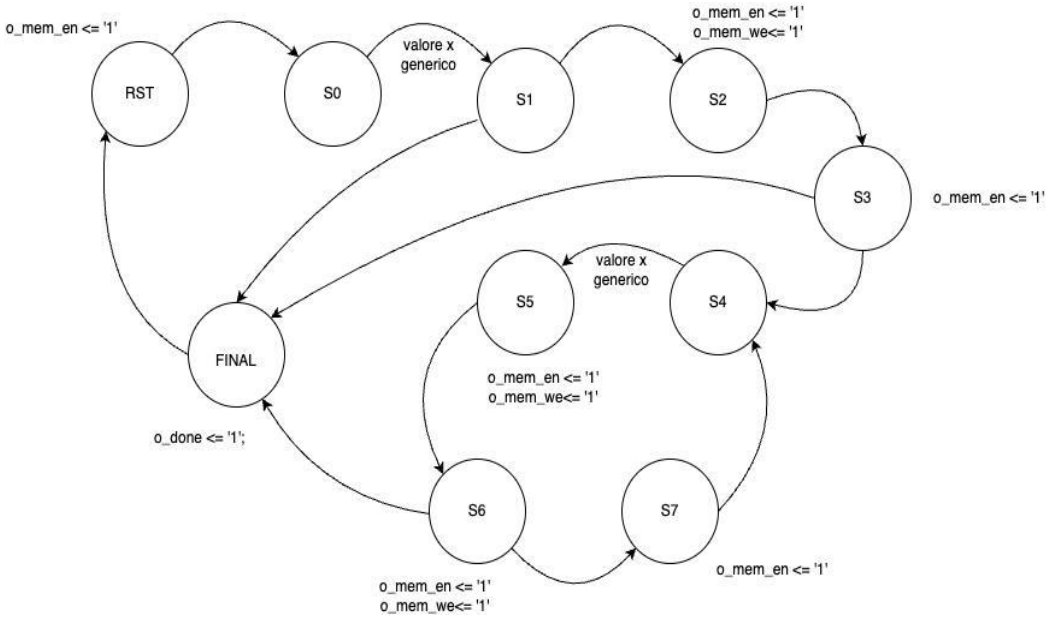


Figura 4

Descrizione degli Stati

1. **rst (reset)**: Questo stato rappresenta la fase di inizializzazione della macchina, innescata dal segnale di reset (i_rst). In questo stato vengono inizializzati tutti i segnali e la macchina è pronta per iniziare la codifica. Rimane in attesa fino a quando il segnale i_start diventa 1, avviando così la prima richiesta di lettura in memoria all'indirizzo i_add.
2. **s0 (attesa della RAM)**: In questo stato, la macchina attende il dato letto, preparandosi per il successivo stato (s1). Nel frattempo, imposta il valore di k = i_k e passa allo stato finale qualora k fosse uguale a zero.
3. **s1 (lettura del primo valore W)**: La macchina legge il primo valore W, lo salva e verifica se è uguale a zero. In caso affermativo, azzererà la credibilità.
4. **s2 (prima scrittura in memoria)**: In questo stato, la macchina sostituisce lo zero in memoria con il valore della credibilità. Se la sequenza è terminata, passa allo stato finale; altrimenti, prosegue nello stato s3.
5. **s3 (richiesta del secondo valore W)**: La macchina richiede il secondo valore W, incrementando l'indirizzo o_mem_addr.
6. **s4 (attesa RAM)**: La macchina attende il dato letto, preparandosi per il successivo stato (s5).

7. **s5 (lettura del valore W)**: Legge il valore e, se è uguale a zero, scrive il valore precedentemente salvato, altrimenti salva il nuovo.
8. **s6 (scrittura in memoria)**: Sovrascrive il valore zero in memoria con il valore di credibilità valore e passa allo stato successivo o allo stato finale se la sequenza è terminata.
9. **s7 (richiesta del valore W)**: La macchina richiede il nuovo valore e ritorna allo stato s4 di attesa.
10. **final**: Indica il completamento della codifica. Imposta o_done a uno e rimane in attesa fino a quando non riceve i_start uguale a zero. Successivamente, si sposta nuovamente nello stato di reset (rst) per prepararsi per una nuova codifica.

2.3 Segnale di Reset

Nel caso in cui fosse ricevuto un segnale di reset in un determinato istante, si procederebbe con l'inizializzazione della macchina, la quale quindi permarrebbe in uno stato di attesa per una successiva codifica.

3 Risultati Sperimentali

3.1 Report Utilization

Nel primo report vengono evidenziati lo spazio occupato e le tipologie di componenti sintetizzate. Nella realizzazione del progetto si è prestata attenzione nell'evitare l'utilizzo di latch.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	86	0	134600	0.06
LUT as Logic	86	0	134600	0.06
LUT as Memory	0	0	46200	0.00
Slice Registers	73	0	269200	0.03
Register as Flip Flop	73	0	269200	0.03
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

3.2 Timing Report

Il secondo report riguarda la simulazione delle tempistiche che tengono conto dei possibili ritardi.
Si nota che l'implementazione rispetta la specifica, infatti prendendo in considerazione il dato MET si evince che il ritardo massimo è 4.072 ns.

```
Timing Report

Slack (MET) :             15.928ns  (required time - arrival time)
  Source:                 state_reg[2]/C
                            (rising edge-triggered cell FDCE clocked by clock  {rise@0.000ns fall@10.000ns period=20.000ns})
  Destination:            C_reg[0]/CE
                            (rising edge-triggered cell FDPE clocked by clock  {rise@0.000ns fall@10.000ns period=20.000ns})
  Path Group:             clock
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            20.000ns  (clock rise@20.000ns - clock rise@0.000ns)
  Data Path Delay:        3.690ns  (logic 0.999ns (27.073%)  route 2.691ns (72.927%))
  Logic Levels:           3  (LUT4=1 LUT6=2)
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.077ns = ( 22.077 - 20.000 )
    Source Clock Delay      (SCD):    2.400ns
    Clock Pessimism Removal (CPR):    0.178ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
```

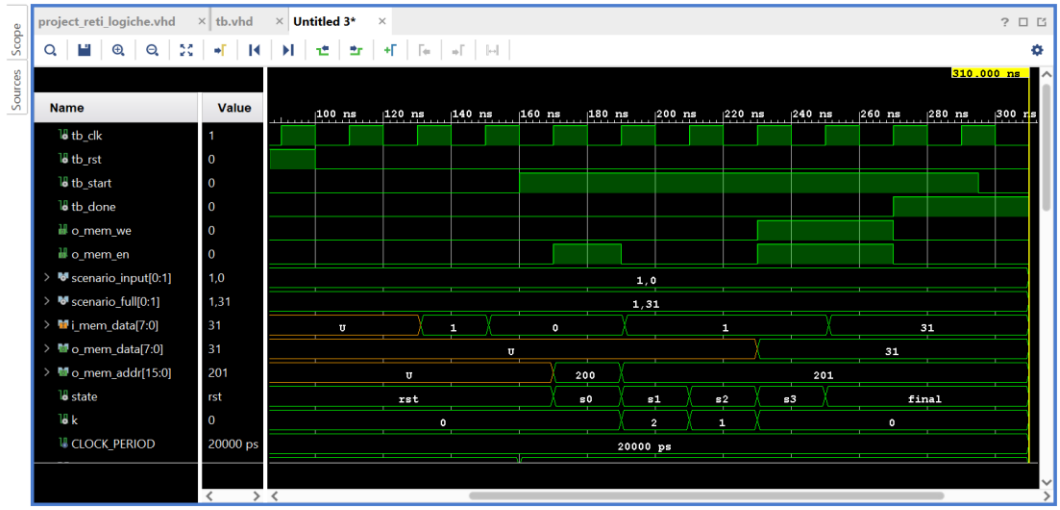
3.3 Testing

In fase iniziale, il nostro progetto è stato soggetto a una serie di test conformi ai requisiti specificati, volti a garantire che:

- Il valore di credibilità non si riducesse al di sotto del limite zero.
- I valori estremi delle parole (0 e 255) non generassero errori né all'inizio né all'interno di una sequenza.
- Le sequenze prive di decremento del valore di credibilità non risultassero in errori.

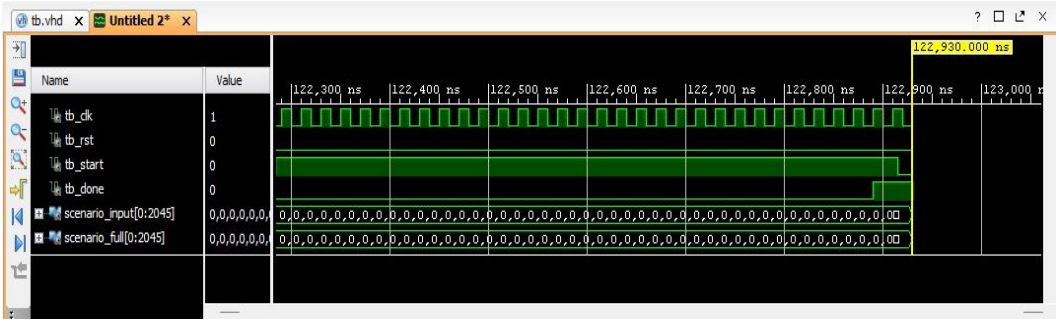
Successivamente, abbiamo sviluppato ulteriori casi di test al fine di esaminare situazioni limite non precedentemente affrontate.

1. **Testbench 1**: Sequenza minima



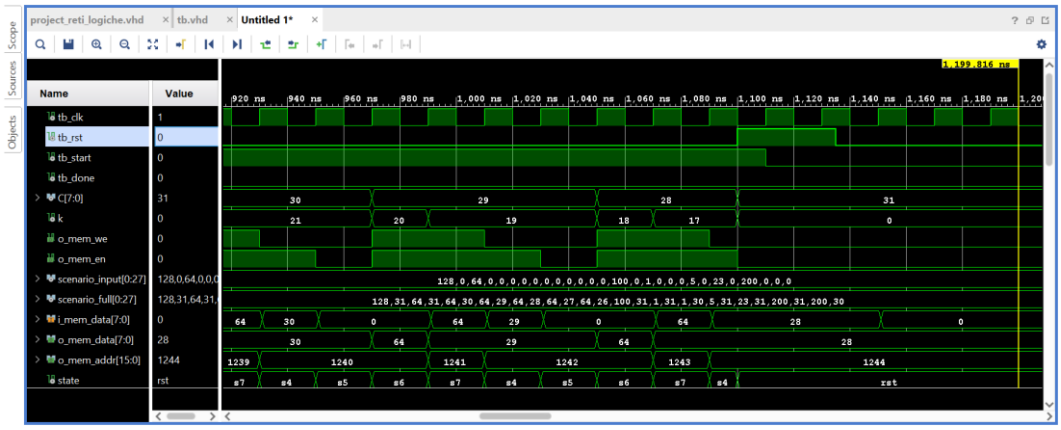
In questo caso abbiamo testato la macchina con una sequenza formata solamente da due valori, in quanto caso limite non rilevando errori.

2. **Testbench 2**: Sequenza massima di soli zero



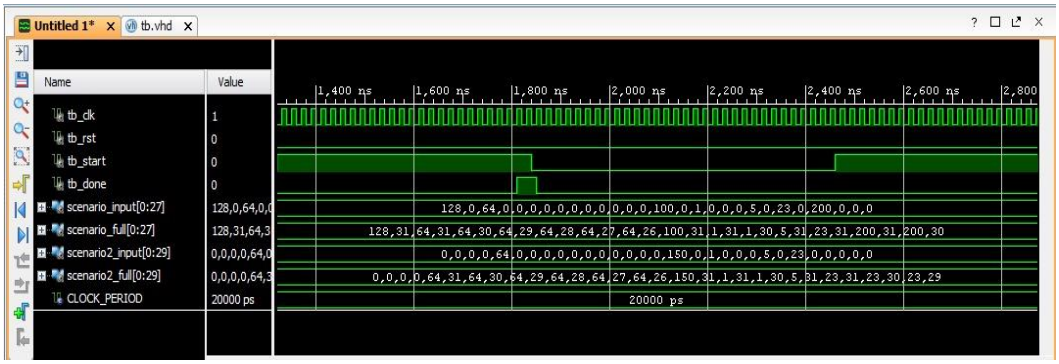
Abbiamo testato anche il caso opposto, ovvero la sequenza massima, oltretutto abbiamo verificato anche il caso di una sequenza di soli zero e anche in questo caso la macchina ha risposto correttamente.

3. **Testbench 3**: segnale di reset



In questo test viene verificato il corretto comportamento della macchina in caso di segnale i_rst alzato in un qualsiasi momento.
Si può notare come l'FSM sia stata riportata nello stato di reset, la credibilità (C) sia stata riportata a 31 e K di nuovo uguale a zero.

4. **Testbench 4**: seconda elaborazione



In questo testbench viene verificato che la macchina sia in grado di elaborare più sequenze di parole.
Si può notare dall'immagine che alla fine della prima sequenza il segnale o_done viene alzato, subito dopo i_start viene abbassato e l'FSM viene portata nello stato di reset, in attesa della sequenza successiva.

4 Conclusioni

Il presente progetto ha offerto un'opportunità significativa di acquisire esperienza pratica nel settore della programmazione hardware tramite l'impiego del linguaggio VHDL. Attraverso l'elaborazione e l'attuazione di diverse componenti hardware, è stato possibile sviluppare una comprensione approfondita dei principi fondamentali della progettazione dei circuiti digitali, nonché della gestione del software Vivado. Tale iniziativa ha contribuito non soltanto alla solidificazione delle conoscenze teoriche precedentemente acquisite nel corso degli studi, bensì ha altresì catalizzato il nostro ingegno creativo e le nostre capacità di risoluzione dei problemi.