Sistema Morse su FPGA: progetto di codifica e decodifica

Giulia Colasanti, Armando Palermo

***Abstract*— Il seguente lavoro presenta l'implementazione di un sistema di codifica e decodifica del codice Morse utilizzando il dispositivo logico programmabile FPGA (Field Programmable Gate Array) Artix-7 di Xilinx, implementato sulla scheda Basys 3. L'obiettivo principale è stato quello di sfruttare appieno i componenti hardware della scheda, come switch, LED, push button e display a 7 segmenti, per creare un flusso integrato e funzionale che unisse input, elaborazione e output. Questo lavoro mostra come le FPGA possano essere utilizzate per applicazioni semplici ma utili, contribuendo a migliorare l'accessibilità e l'inclusione in ambiti tecnologici e comunicativi.Un elemento aggiuntivo, infatti, è il suo possibile utilizzo come strumento di supporto per persone con disabilità motorie. La codifica del codice Morse è resa accessibile attraverso segnali visivi, come l'illuminazione dei LED e la visualizzazione delle lettere e dei numeri sul display a 7 segmenti, offrendo un'alternativa alla tradizionale trasmissione sonora.**

I. Introduzione

Il codice Morse [1], sviluppato nel XIX secolo da Samuel Morse e Alfred Vail, ha rappresentato una pietra miliare nella storia della comunicazione, permettendo la trasmissione di messaggi tramite impulsi di lunghezza variabile, che corrispondono a punti e linee. Utilizzato principalmente nel telegrafo, divenne rapidamente lo standard per le comunicazioni a lunga distanza, in particolare in ambito marittimo, e rimase in uso fino al 1999. Il 24 maggio 1844, il primo messaggio telegrafico, *"What hath God wrought?" (Cosa ha fatto Dio?)*, fu inviato sulla linea telegrafica tra Baltimora e Washington, segnando un momento cruciale nell'evoluzione della comunicazione. Nonostante l'alfabeto Morse sia stato in gran parte sostituito da tecnologie più moderne nella comunicazione quotidiana, continua a essere utilizzato in contesti specifici, come nel settore medico, grazie alla sua semplicità e affidabilità. Quest'ultimo si rivela particolarmente utile per i pazienti affetti da Malattia del Motoneurone (MND) [2], poiché richiede solo movimenti minimi, come quelli delle dita, per comunicare. Con il progredire della malattia, i pazienti spesso perdono la capacità di parlare, ma continuano a mantenere un certo controllo motorio nelle mani, rendendo il codice Morse una soluzione ideale per la comunicazione.Recenti sviluppi nelle tecnologie IoT hanno reso possibile tracciare il movimento delle dita in maniera semplice ed efficace, senza bisogno di dispositivi complessi o costosi, come nel caso del tracciamento oculare utilizzato da Stephen Hawking. In questo contesto, l' FPGA può essere coinvolta come dispositivo di supporto per la tramutazione, in codice morse, dei segnali provenienti dai minimi movimenti dei pazienti.La sequenza di punti e linee verrà successivamente tradotta in un testo semplificando così la comunicazione. Il progetto in questione ha come obiettivo la realizzazione di un sistema che, sfruttando la capacità di elaborazione dell'FPGA Artix-7 della Xilinx, permetta di decodificare e codificare codice Morse in caratteri alfanumerici (lettere e numeri) e viceversa.In pratica, il sistema agirà come un encoder, che traduce la sequenza di punti e linee, presa in ingresso, in caratteri alfanumerici, e come un decoder, che interpreta un carattere alfanumerico e lo converte nel formato Morse. La realizzazione del sistema è stata possibile grazie all'uso del software Vivado [3], uno strumento essenziale utile alla progettazione e simulazione di circuiti digitali, che permette di scrivere, testare e sintetizzare codice VHDL (Very High Speed Integrated Circuit Hardware Description Language) [4], tramite il quale si possono descrivere circuiti digitali a basso livello.Il risultato finale è un encoder-decoder che sfrutta esclusivamente i componenti integrati della piattaforma Basys 3, senza necessitare di periferiche esterne.
I contributi principali di questo progetto sono:

- Traduzione Bidirezionale del Codice Morse: Implementazione di un sistema in grado di codificare caratteri alfanumerici in codice Morse e di decodificare il codice Morse nei corrispondenti caratteri o numeri;
- Utilizzo delle risorse hardware dell'FPGA: è stata utilizzata un'interfaccia utente semplice e intuitiva, che consente di sfruttare unicamente le risorse messe a disposizione dalla piattaforma, permettendo all'utente di usare come input gli switches e i pushbuttons. Il sistema è progettato per gestire un singolo carattere o numero alla volta, semplificando la visualizzazione sul display a 7 segmenti.Questo approccio elimina la necessità di periferiche esterne;
- Uso del linguaggio VHDL: Il progetto rappresenta un esempio pratico e accessibile a tutti, dimostrando le

capacità del linguaggio VHDL nel progettare sistemi che elaborano segnali in tempo reale e enfatizzando l'importanza di una progettazione basata sull'hardware.Il sistema realizzato infatti elabora il codice Morse in tempo reale, garantendo che la codifica e la decodifica avvengano quasi istantaneamente;
- Scalabilità: questa implementazione funge da base per lo sviluppo di sistemi di comunicazione avanzati.L'architettura del sistema è scalabile in quanto organizzata in moduli separati, permettendone una facile estensione;

Il presente documento è organizzato come segue. Nella Sezione 2 vengono analizzati i lavori correlati. Nella Sezione 3 viene descritta l'implementazione del progetto. Nella Sezione 4 sono mostrati i risultati sperimentali e nella Sezione 5 le conclusioni.

II. Lavori correlati

Nell'ambito della codifica/decodifica del codice Morse, diversi studi si sono focalizzati sull'uso delle FPGA in sistemi di comunicazione accessibili, come strumenti di supporto, a persone con disabilità motorie. In questa sezione, verranno descritti due studi che hanno fornito ispirazione e spunti per lo sviluppo del nostro progetto. Attraverso l'analisi di questi lavori, abbiamo compreso che la codifica e la decodifica Morse continuano a mantenere una notevole utilità anche oggi, nonostante non venga più utilizzata con la stessa diffusione di un tempo. Nell'articolo *"FPGA Based Morse Code Communicator for Visual and Speech Impaired People using Basys-3"* [5], il sistema progettato ha specifiche molto simili a quelle del nostro progetto, in particolar modo per quanto riguarda la parte di decodifica da Morse a carattere/numero.Anche in questo caso è stato utilizzato il display a 7 segmenti per la visualizzazione del risultato. Dall'analisi, emerge che la gestione del clock nel loro sistema è stata implementata in maniera manuale, con l'utente responsabile dell'attivazione e disattivazione del clock tramite un input fisico. Nel nostro caso, invece, abbiamo utilizzato un clock automatico, ovvero il clock nativo dell'FPGA. Il vantaggio di questa scelta risiede nella capacità del clock automatico di fornire una temporizzazione stabile e coerente, senza la necessità dell'intervento umano.Inoltre, abbiamo considerato anche la gestione del debouncing dei pulsanti, un aspetto che non è stato affrontato nello studio precedente. Il debouncing è fondamentale per evitare che il sistema interpreti erroneamente i segnali di ingresso a causa dei rimbalzi elettrici dei pulsanti, migliorando così l'affidabilità e la precisione del sistema.
Nel secondo articolo *"FPGA-based communication interface for persons with motor neuron diseases"*, [6] lo scopo principale rimane il medesimo, tuttavia l'argomento viene affrontato in maniera più sofisticata e approfondita. Il cuore del sistema sono i sensori MEMS (micro electro-mechanical systems) utilizzati per rilevare i micro movimenti muscolari. Il segnale attivato dal sensore 1 è interpretato come punto, mentre il segnale dal sensore 2 come una linea. Questi vengono inviati all'FPGA che li elabora, eliminando eventuali rumori e segnali indesiderati. I caratteri decodificati vengono poi visualizzati su un display LCD. Concludendo quindi, come dimostra il secondo articolo, la decodifica Morse trova ancora oggi applicazioni concrete nella realtà, tramite sistemi basati su FPGA. Il nostro progetto rimane comunque un sistema di base, sviluppato con uno scopo puramente didattico con l'obiettivo di apprendere i principi della progettazione su FPGA.

III. Descrizione del progetto

L'obiettivo del progetto è la realizzazione di un codificatore/decodificatore Morse. Il sistema è diviso in due modalità:

- Modalità di codifica: dove il sistema converte i caratteri alfanumerici in codice Morse
- Modalità di decodifica: dove il sistema interpreta i segnali Morse e li converte in caratteri o numeri.

Per quanto riguarda la gestione del clock, si è scelto di non utilizzare quello nativo dell'FPGA. Al suo posto, tramite un modulo, `Divisore_clk`, è stato generato un clock da 10 MHz. Questa scelta è stata adottata con l'obiettivo di limitare il consumo di potenza dinamico, come verrà descritto nella sezione dedicata ai risultati.

A. Codifica

Questa modalità è divisa in 2 moduli: `Mapping_SW_Morse` ed `Encoder_LED`. L'idea è quella far inserire all'utente un carattere alfanumerico in formato binario, che successivamente verrà tradotto in codice Morse dal primo modulo. Tale codice viene usato per far lampeggiare un LED dal secondo modulo.

1) Mapping_SW_Morse: Il modulo prende in input un vettore di 8 bit, *switchIn* (STD_LOGIC_VECTOR), il cui valore è definito da 8 switch presenti sulla scheda Basys 3. La configurazione degli switch rappresenta il carattere o numero in formato binario.

$$es.\ D \rightarrow 01000100$$

La logica è definita in un processo che viene eseguito ogni volta che l'input cambia. Al suo interno tramite una struttura *case* il valore di switchIn viene confrontato con i possibili valori, binari identificativi dei caratteri che vanno da A-Z e dei numeri da 0-9. Se switchIn corrisponde ad uno dei valori definiti nel case il modulo assegna all'output il relativo codice Morse(in formato binario), altrimenti viene assegnato un valore di default.
L'uscita *mappedMorse* è un vettore a 16 bit in binario dove '1' corrisponde ad un punto , '11' a una linea e '0' sono gli spazi tra i simboli

$$es.\ D \rightarrow 01000100 \rightarrow 1101010000000000 \rightarrow \text{- . .}$$

La sua dimensione è legata alla necessità di gestire correttamente gli spazi tra le sequenze, di mantenere una lunghezza uniforme per tutti i caratteri Morse e di regolare correttamente la temporizzazione dell'accensione del LED attraverso il modulo *Encoder_LED*.

2) Encoder_LED: Lo scopo del modulo è l'accensione e lo spegnimento ad intermittenza del LED in base al carattere/numero Morse che si deve rappresentare. L'input, *morse_code*(STD_LOGIC_VECTOR a 16 bit) , rappresenta una sequenza di codifica Morse ottenuta dal modulo precedente.
Il comportamento principale avviene all'interno di un processo, attivato dal segnale di clock.
Morse_code viene scandito bit per bit. Quando il bit è '1' il led viene acceso per un tempo prestabilito, se invece è '0' il led resta spento per un diverso periodo prefissato. La temporizzazione viene gestita attraverso due variabili *wait_timer_on o wait_timer_off*, utilizzate come contatori. I loro valori massimi sono stati definiti sulla base di un clock di 10 MHz, e sono stati calcolati secondo la seguente formula.

$$Timer = \frac{DurataDesiderata(secondi)}{DurataCiclo(secondi)} \quad (1)$$

dove

$$DurataCiclo = \frac{1}{FrequenzaClock(Hz)} \quad (2)$$

Sono stati decisi i seguenti tempi:

- durata accensione LED per un punto: 0,5s (Timer : 5 mln di cicli)
- durata accensione LED per una linea: 1s (Timer : 10 mln di cicli)
- durata spegnimento LED per uno spazio: 0,3s (Timer: 3 mlm di cicli)

Se il bit letto è 1 viene incrementato wait_timer_on ad ogni ciclo di clock e quando raggiunge il valore limite (5 mln di cicli) il LED si spegne, il timer è resettato e si passa al controllo del successivo bit; se il bit è ancora '1', il LED si riaccende per un altro periodo di 0,5s. In caso contrario, il LED rimane spento fino a quando la variabile wait_timer_off non raggiunge il valore massimo (3 mln di cicli). Per rappresentare una linea, il LED viene trattato come se si accendesse per due periodi consecutivi di 0,5 secondi, separati da uno spegnimento della durata di un singolo ciclo di clock (100 nanosecondi). Questo spegnimento, essendo così breve, risulta impercettibile all'occhio umano, creando l'impressione che il LED rimanga acceso ininterrottamente per un intero secondo.

B. Decodifica

Questa modalità è divisa in 3 moduli: `Decoder_SW_Morse`, `DebounceControlModule` e `Segment_Display`. In questa situazione l'utente, tramite i pushbutton, compone la sequenza Morse che viene successivamente decodificata nel carattere/numero corrispondente sul display a 7 segmenti.

1) Decoder_SW_Morse: Questo è il modulo principale che interpreta i segnali in ingresso, *pushIn* (STD_LOGIC) traducendoli in codice binario, *mappedBinary* (STD_LOGIC_VECTOR di 16 bit) per rappresentare i punti e le linee.
La logica prevede un processo che viene attivato ad ogni ciclo di clock. Ad ogni fronte di salita del clock se pushIn = 1, viene incrementato un contatore, *pushTimerON*, che misura la durata del tempo in cui il pulsante rimane premuto. Quando è rilasciato, si incrementa *pushTimerOFF*. Successivamente sarà aggiunto il simbolo Morse corrispondente, tramite un controllo su *pushTimerON*:

- se pushTimerON $>$ 5 mln (0,5s), corrisponde ad una linea e al vettore d'uscita viene aggiunto '110'
- se pushTimerON $<$ 5 mln (0,5s), corrisponde ad un punto ed al vettore d'uscita viene aggiunto '10'.

Lo zero sta ad indicare lo spazio tra i simboli. Questo è coerente con la struttura del vettore utilizzato per rappresentare il codice Morse, come descritto in precedenza (**vedi II.2**).
Per differenziare le varie sequenze di simboli e quindi i caratteri, il processo prevede un ulteriore controllo che tiene conto del tempo di inattività del pulsante. Dopo 0,7 secondi di inattività la sequenza è considerata finita.

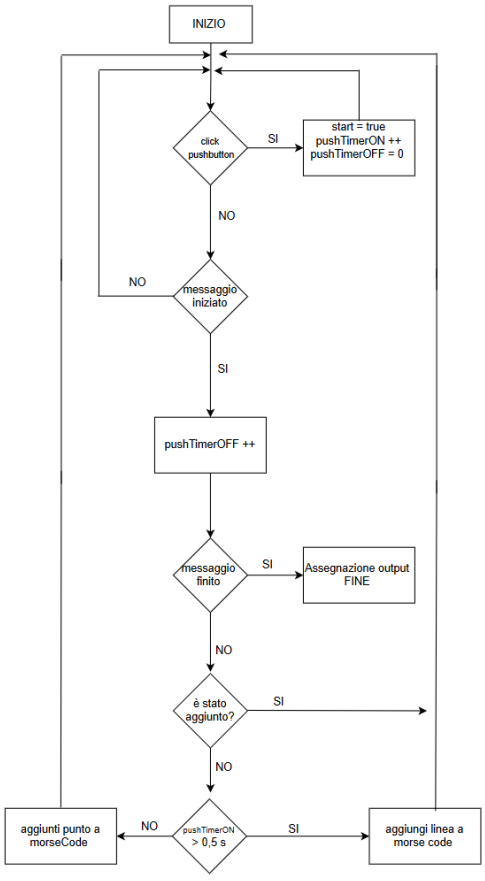

Fig. 1. Flow Chart del modulo Decoder_SW_Morse

2) DebounceControlModule: L'utilizzo dei pushbutton può portare ad effetti indesiderati dovuti a dei rimbalzi (bounce) , ovvero delle oscillazioni che si verificano quando il pulsante viene premuto o rilasciato, causate dalla sua natura meccanica [7]. Il sistema quindi potrebbe interpretare erroneamente una singola pressione del pulsante come multiple.
Lo scopo del seguente modulo è quello di filtrare i rimbalzi del pulsante, garantendo così la stabilità del segnale di uscita (*btn_out*). Il contatore (*counter*) viene utilizzato per verificare che il pulsante rimanga stabile per un periodo di tempo specificato (20 ms) prima di aggiornare lo stato stabilizzato (*btn_reg*). Se il pulsante cambia stato, il contatore viene resettato, altrimenti viene incrementato fino a raggiungere il valore massimo (*DEBOUNCE_COUNT_MAX*). Solo quando il contatore raggiunge questo valore, lo stato del pulsante viene considerato stabile e aggiornato.

```
    process(clk)
    begin
        if clk'event and clk = '1' then
            -- Se il pulsante è cambiato rispetto allo stato precedente
            if btn_in /= btn_past then
                -- Se il pulsante è cambiato, resetta il contatore
                counter <= 0;
                btn_reg <= btn_in;  -- Aggiorna la registrazione del pulsante
            else
                -- Se il pulsante è stabile, incrementa il contatore
                if counter < DEBOUNCE_COUNT_MAX then
                    counter <= counter + 1;
                else
                    -- Dopo il tempo di debounce, aggiorna lo stato finale
                    btn_reg <= btn_in;
                end if;
            end if;
            -- Memorizza lo stato precedente del pulsante
            btn_past <= btn_in;
        end if;
    end process;
```

Fig. 2. Processo del modulo DebounceControlModule

3) Segment_Display: La logica è simile a quella analizzata nel modulo Mapping_SW_Morse (**vedi II.2**). In base a un input binario Morse (un vettore di 16 bit), ottenuto dal Decoder_Morse, viene mappata una sequenza specifica di bit per indicare quali segmenti devono essere accesi per visualizzare correttamente il carattere associato.

$$es.\ ”1101010000000000” \rightarrow 1000010 \rightarrow d$$

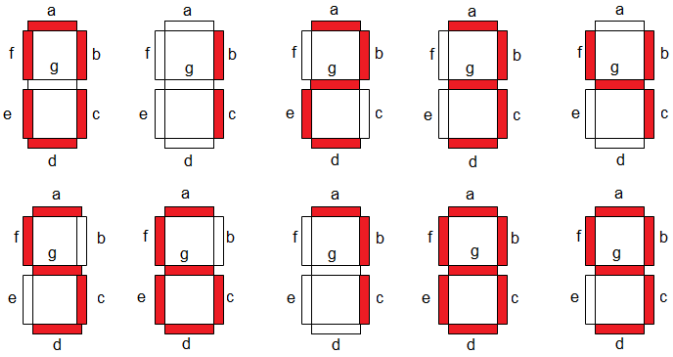

Fig. 3. Funzionamento Display a 7 segmenti

Se il valore di binaryMorse non corrisponde a un valore mappato, tutti i segmenti vengono spenti.Il controllo degli anodi è fisso (”1110”), il che significa che solo un display (il primo) è attivo.

I moduli elencati vengono aggregati in un unico modulo definito TOP_levelModule. Ognuno di essi è instanziato come componente e sono messi in comunicazione tramite un port mapping. Il *port mapping* [8] in VHDL è il processo mediante il quale si collegano i segnali interni del modulo di livello superiore (o del sistema) alle porte di input e output dei moduli instanziati, stabilendo così la comunicazione tra di essi.

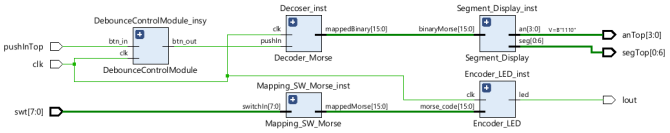

Fig. 4. Schema RTL modulo TOP_levelModule

IV. Risultati

Come discusso nelle sezioni precedenti, per la realizzazione di questo progetto è sufficiente utilizzare esclusivamente le risorse offerte dalla scheda Basys 3, che integra un FPGA della famiglia Artix-7 di Xilinx. Il sistema implementato supporta due modalità operative differenti a seconda degli input attivati , switch o pushbutton , producendo di conseguenza output distinti, rappresentati rispettivamente dai LED o dal display a 7 segmenti.

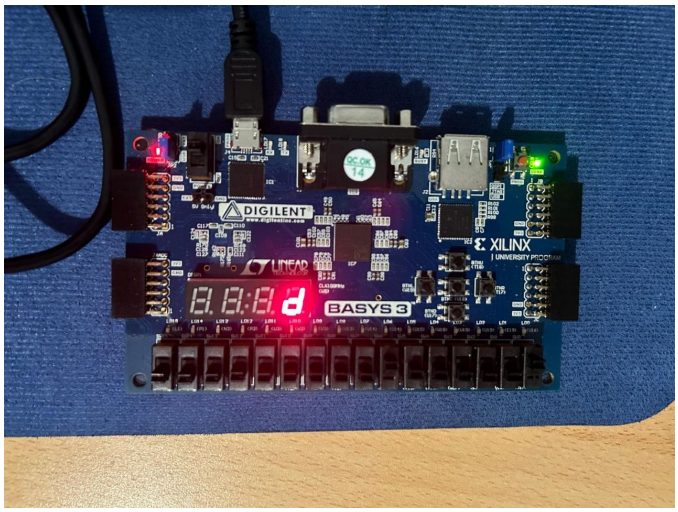

Fig. 5. Decodifica lettera d su FPGA

Tramite il file di constrait , abbiamo definito:

- per codifica 8 switch, V16 a W13 , e il LED mappato su pin U16;
- per decodifica il pushbutton centrale corrispondente al pin U18, i pin da W7 a U7 per il display e gli anodi da U2 a W4, configurati in maniera tale da visualizzare solo cifra più a destra;
- la temporizzazione viene gestita da un clock generato (10 MHz) partendo da quello nativo (100MHz) mappato sul pin W5;

In figura si riporta l'intera configurazione

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property PACKAGE_PIN V17 [get_ports swt[0]]
set_property IOSTANDARD LVCMOS33 [get_ports swt[0]]
set_property PACKAGE_PIN V16 [get_ports {swt[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {swt[1]}]
set_property PACKAGE_PIN W16 [get_ports {swt[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {swt[2]}]
set_property PACKAGE_PIN W17 [get_ports {swt[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {swt[3]}]
set_property PACKAGE_PIN W15 [get_ports {swt[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {swt[4]}]
set_property PACKAGE_PIN V15 [get_ports {swt[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {swt[5]}]
set_property PACKAGE_PIN W14 [get_ports {swt[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {swt[6]}]
set_property PACKAGE_PIN W13 [get_ports {swt[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {swt[7]}]

## LEDs
set_property PACKAGE_PIN U16 [get_ports lout]
set_property IOSTANDARD LVCMOS33 [get_ports lout]

##7 segment display
set_property PACKAGE_PIN W7 [get_ports {segTop[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {segTop[0]}]
set_property PACKAGE_PIN W6 [get_ports {segTop[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {segTop[1]}]
set_property PACKAGE_PIN U8 [get_ports {segTop[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {segTop[2]}]
set_property PACKAGE_PIN V8 [get_ports {segTop[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {segTop[3]}]
set_property PACKAGE_PIN U5 [get_ports {segTop[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {segTop[4]}]
set_property PACKAGE_PIN V5 [get_ports {segTop[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {segTop[5]}]
set_property PACKAGE_PIN U7 [get_ports {segTop[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {segTop[6]}]
#ANODS
set_property PACKAGE_PIN U2 [get_ports {anTop[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {anTop[0]}]
set_property PACKAGE_PIN U4 [get_ports {anTop[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {anTop[1]}]
set_property PACKAGE_PIN V4 [get_ports {anTop[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {anTop[2]}]
set_property PACKAGE_PIN W4 [get_ports {anTop[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {anTop[3]}]

##Buttons
set_property PACKAGE_PIN U18 [get_ports pushInTop]
set_property IOSTANDARD LVCMOS33 [get_ports pushInTop]
```

Fig. 6. Configurazione file .xdc

Di seguito vengono forniti i dati relativi alle risorse utilizzate dall'FPGA dopo l'implementazione e i dati legati al consumo di potenza.

Resource	Utilization	Available	Utilization %
LUT	240	20800	1.15
FF	183	41600	0.44
IO	22	106	20.75

Fig. 7. Risorse utilizzate

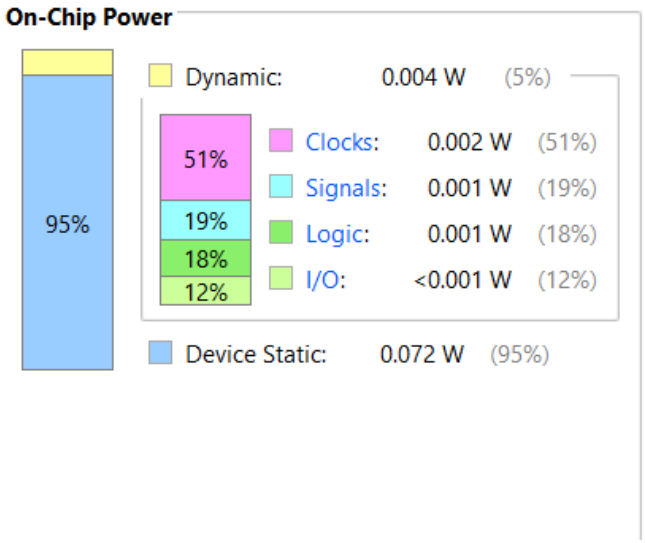

Fig. 8. Consumo di potenza a 100 Mhz

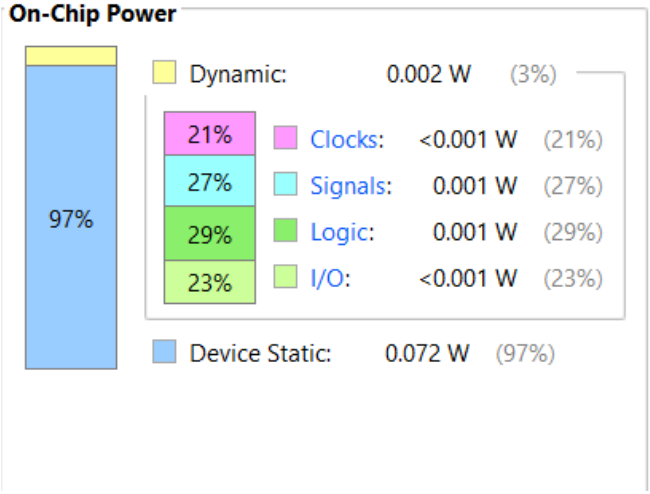

Fig. 9. Consumo di potenza a 10 Mhz

In figura 8 la frequenza di clock utilizzata è 100MHz, ovvero quella nativa dell'FPGA utilizzata. La potenza statica costituisce la parte predominante del consumo energetico,, mentre la potenza dinamica è molto contenuta. Il principale contributo di quest'ultima è dato dal clock(51%).
In figura 9 la frequenza è stata ridotta dal modulo Divisore_clk a 10 MHz. Sebbene la potenza statica resti il contributo maggiore, quella dinamica è diminuita. Ciò è dovuto al fatto che, diminuendo la frequenza di clock, aumenta il tempo durante il quale il segnale rimane stabile (livello alto o basso), rendendo le transizioni meno frequenti.

V. Conclusioni

I test effettuati hanno dimostrato il corretto funzionamento di entrambe le modalità. Il sistema risponde coerentemente agli input, gestendo eventuali rimbalzi meccanici dei pushbutton. Il progetto è stato sviluppato con scopi puramente didattici, ma sono certamente possibili ulteriori miglioramenti, come la connessione dell'FPGA con dispositivi esterni tramite UART per gestire l'inserimento di un intero testo e la rappresentazione di caratteri che non possono essere visualizzati su un display a 7 segmenti. Inoltre, sarebbe possibile ridurre ulteriormente il consumo dinamico limitando l'attività del clock nei moduli non attivi.

References

[1] Wikipedia. Wikipedia-Codice Morse, Accessed November 2024. [https://it.wikipedia.org/wiki/Codice$_Morse$].

[2] Yuhang Jiang Jie Xiong Qin Lv Youwei Zeng Daqing Zhang Kai Niu, Fusang Zhang. Wimorse: a contactless morse code text input system using ambient wifi signals, 20 Oct 2019. [https://dx.doi.org/10.1109/JIOT.2019.2934904].

[3] AMD. Vivado software, Accessed December 2024. [https://www.xilinx.com/support/download.html].

[4] Elettronica Open Source. Elettronica Open Source - Linguaggio VHDL, Accessed December 2024. [https://it.emcelettronica.com/il-linguaggio-vhdl].

[5] R. Seetharaman; M. Ramajayam; K. Kamalakannan; M. Vignesh; R.L. Saran; K. Anandan. FPGA Based Morse Code Communicator for Visual and Speech Impaired People using Basys-3, 2022. [https://doi.org/10.1109/ICEARS53579.2022.9752384].

[6] Zbigniew Hajduk. FPGA-based communication interface for persons with motor neuron diseases, 2016. [https://doi.org/10.1016/j.bspc.2016.01.007].

[7] Wikipedia. Wikipedia-Circuito antirimbalzo, Accessed December 2024. [https://it.wikipedia.org/wiki/Circuito$_anti-rimbalzo$].

[8] New York City College of technology. VHDL COMPONENTS AND PORT MAP, Accessed December 2024. [https://openlab.citytech.cuny.edu/wang-cet4805/files/2020/02/VHDL-Components-and-Port-Maps.pdf].