Politecnico di Milano
Prova Finale A.A. 2021/2022
Progetto di Reti Logiche

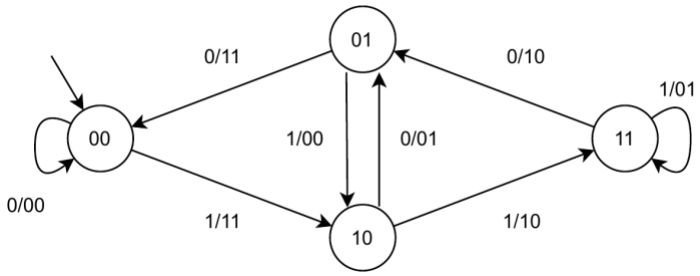
Pierluigi Negro c.p. 10670080
Marco Molè c.p. 10676087

Referente: Prof. Fabio Salice

Indice

1 Specifica

Il progetto consiste nell'implementare un codificatore convoluzionale con tasso di trasmissione $\frac{1}{2}$ che si interfacci con una semplice memoria contente parole da 8 bit. Il linguaggio di descrizione dell'hardware usato è VHDL. Il codificatore convoluzionale, d'ora in poi CC, è rappresentabile con la seguente macchina di Moore.



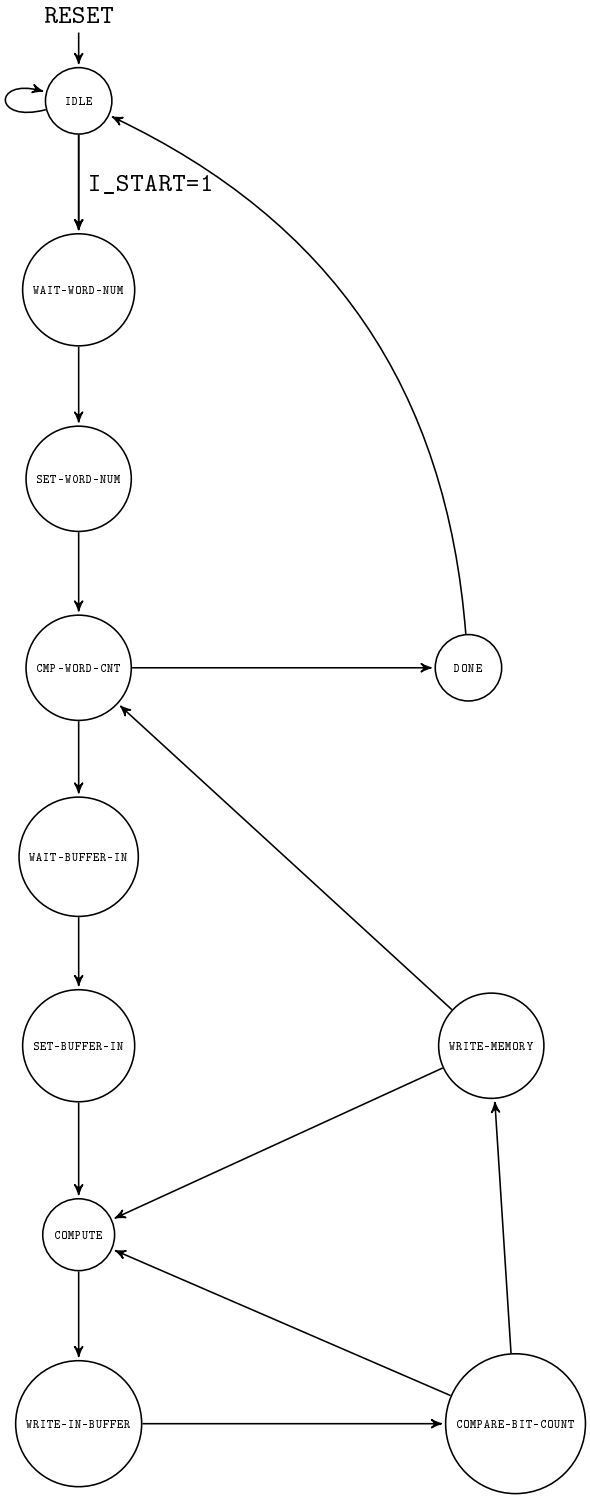
Siccome il CC ha in input e in output degli stream di bit, il modulo HW dovrà occuparsi della serializzazione e de-serializzazione dei due flussi. In particolare il flusso in uscita verrà de-serializzato con un concatenamento alternato.

INSERIRE ESEMPIO

Protocollo tra il modulo e la memoria L'elaborazione inizia quando il segnale di START in ingresso viene portato a 1 e rimane tale per tutta la durata della elaborazione. Una volta scritta in memoria l'ultimo byte il modulo alza il segnale DONE che segnala il termine della elaborazione. Il segnale DONE deve rimanere a 1 fino a che il test bench non porta START a 0. Il test bench non può dare un altro segnale di START fino a che DONE non è stato portato a zero. Il protocollo prevede quindi che il modulo debba supportare la codifica di più flussi uno dopo l'altro. Ad ogni nuovo flusso lo stato del CC viene resettato allo stato di partenza. Il RESET è necessario solo per la prima elaborazione, dato che dalla seconda in poi basterà rispettare il protocollo appena descritto

2 Scelte progettuali

Il modulo è stato pensato come una macchina a stati finiti con reset asincrono. Per la gestione del numero delle parole da elaborare vengono usati due registri WORD_NUMBER e WORD_COUNTER.
Nel primo viene copiato il byte nella posizione 0 che specifica quante parole sono da elaborare. Il secondo viene incrementato ad ogni parola elaborata.
La serializzazione avviene leggendo un byte alla volta, in modo da rendere le operazioni di lettura e scrittura in memoria più semplici.


RESET
IDLE
I_START=1
WAIT-WORD-NUM
SET-WORD-NUM
CMP-WORD-CNT
DONE
WAIT-BUFFER-IN
SET-BUFFER-IN
WRITE-MEMORY
COMPUTE
WRITE-IN-BUFFER
COMPARE-BIT-COUNT

2.1 Descrizione

1. La macchina parte da uno stato di IDLE in cui aspetta il segnale di I_START per iniziare. Arrivato il segnale inizia a preparare la fase di lettura del primo byte della memoria.
    - In questa fase viene inizializzato a zero anche il registro WORD_COUNTER che terrà conto del numero di parole elaborato finora.
2. Legge il primo byte e lo salva nel registro WORD_NUMBER. Questo avviene negli stati WAIT_WORD_NUMBER e SET_WORD_NUMBER.
3. Nello stato COMPARE_WORD_COUNT confronta il contenuto dei registri WORD_NUMBER e WORD_COUNTER.
    - Se ha elaborato tutte le parole porta il segnale O_DONE a 1 e va nello stato DONE in cui aspetta che I_START venga portato a 0 dal test bench.
    - In caso contrario legge la prossima parola di memoria e la copia BUFFER_IN. La fase di caricamento dalla memoria avviene negli stati WAIT_BUFFER_IN e SET_BUFFER_IN.
4. Nello stato COMPUTE avviene la codifica convoluzionale . Viene usato il registro BUFFER_INDEX per tenere traccia di quale bit sta facendo la codifica.
5. Lo stato successivo, WRITE_IN_BUFFER, scrive i bit p1k e p2k nel BUFFER_OUT. La posizione in cui vengono scritti i due bit è determinata dal valore di BUFFER_INDEX considerato in mod 4.
6. In COMPARE_BIT_COUNT si analizza il contenuto di BUFFER_INDEX
    - Se è uguale a 7 o 3 vuol dire che BUFFER_OUT è pieno e deve essere scritto in memoria.
    - In caso contrario si passa al prossimo bit e si ritorna allo stato di COMPUTE
7. La scrittura in memoria avviene nello stato WRITE_MEMORY . In questo stato avviene pure il controllo su BUFFER_INDEX per capire se ha elaborato entrambe le metà di BUFFER_IN. In caso positivo torna nello stato COMPARE_WORD_COUNT, se no torna nello stato COMPUTE per iniziare l'elaborazione della seconda metà della parola.

3 Risultati dei test

In questa sezione vengono riportati i test che abbiamo ritenuto più significativi per la testare la corretta implementazione del modulo. Tutti i test sono stati eseguiti in behavioural, functional post-sintesi e timing post-sintesi.

3.1 Casi limite del numero di parole

Nel caso in cui il primo byte sia tutto a zero il test bench controlla che il modulo non elabori parole di memoria. Questo avviene inizializzando porzioni di memoria con valori pseudocasuali e controllare che, dopo che il segnale O_DONE sia stato portato a 1, non siano cambiate.

3.2 Memoria con più flussi

Questa famiglia di test serve per testare il protocollo per la codifica dei flussi successivi al primo. Sono possibili vari test di questo tipo

4 Risultati della sintesi

La FPGA su cui è stata fatta la sintesi è la Artix-7a200tfbg484-1 .

4.1 Utilizzazione dello spazio

L'assenza di latch nella sintesi indica una corretta scrittura del codice VHDL.

Tipo di elemento	n° usati	% sul totale
LUT	73	0.05
FF	68	0.03
Latch	0	0

4.2 Timing report

La specifica richiede un clock di almeno 100ns. Nel timing report si vede che il *data path delay* è di 4.037 ns. Si può quindi ipotizzare un corretto funzionamento con frequenze di clock fino ai 247 Mhz.

4.3 Risultato dei Test Bench

Dato che il timing report ci ha indicato che il modulo avrebbe potuto funzionare ben sotto i 100ns abbiamo provato ad abbassare il tempo di clock dei test bench per verificare l'ipotesi fatta. Tutti i test provati hanno funzionato fino a 7ns, con alcuni di quelli più semplici, trattasi di quelli senza flussi o segnali di reset, che hanno funzionato anche a 5ns. Ciò conferma che il *data path delay* fornisce una buona stima della massima frequenza di clock.