Prova finale

Progetto di reti logiche

2019-20

POLITECNICO
MILANO 1863

Gabriele Marra - [redacted]

Matteo Miceli - [redacted]

Sommario

Introduzione **3**

Architettura **4**

Macchina a Stati 4

Schema 4

Descrizione Stati 4

Risultati sperimentali **6**

Simulazioni **7**

Test base 7

#1: ADDR appartiene a una WZ 7

#2: ADDR non appartiene a nessuna WZ 7

Test specifici 8

#3: Esecuzioni consecutive 8

#4: Reset asincrono 8

#5: Stress test 9

Conclusioni **10**

Introduzione

Scopo della prova finale di reti logiche è l'implementazione di un componente descritto in VHDL. Il componente ha l'obiettivo di codificare un indirizzo (ADDR) di 7 bit utilizzando un metodo di codifica a bassa dissipazione di potenza denominato "Working Zone" (WZ).

Una WZ è un intervallo di indirizzi di dimensione fissata, 4 in questo specifico caso, di cui viene specificato l'indirizzo base.

Nel caso in cui l'indirizzo appartenga a una delle WZ, la codifica a 8 bit sarà della forma WZ_BIT & WZ_NUM & WZ_OFFSET, ovvero la concatenazione di: WZ_BIT, un bit che indica l'appartenenza ad una delle WZ che sarà 1 nel caso affermativo; WZ_NUM, numero a 3 bit, indica il numero della WZ a cui l'indirizzo appartiene; WZ_OFFSET indica l'offset dell'indirizzo all'interno della WZ utilizzando la codifica one-hot su 4 bit.

Nel caso in cui invece l'indirizzo non appartenga a nessuna WZ verrà trasmesso WZ_BIT & ADDR, ovvero la concatenazione di 0 e l'indirizzo da codificare così com'è (senza alcuna rielaborazione).

In questo progetto assumeremo che gli indirizzi base di tutte le WZ vengano caricati preventivamente in RAM nelle prime 8 posizioni, nella nona posizione ci sarà l'indirizzo da codificare e la decima è dove verrà salvato l'indirizzo codificato.

ESEMPIO

MEMORIA		**COMMENTO**
Indirizzo	**Valore**	
0	2	Indirizzo base WZ 0
1	14	Indirizzo base WZ 1
2	20	Indirizzo base WZ 2
3	30	Indirizzo base WZ 3
4	56	Indirizzo base WZ 4
5	80	Indirizzo base WZ 5
6	88	Indirizzo base WZ 6
7	95	Indirizzo base WZ 7
8	58	ADDR da codificare
9	196	Risultato

In questo esempio viene mostrato un caso in cui l'indirizzo da codificare (58) appartiene ad una delle WZ caricate in memoria, in particolare alla WZ numero 4 (con indirizzo base 56).

Essendo la WZ un intervallo di 4 indirizzi l'indirizzo da codificare ha quindi rispetto alla WZ un offset di 2 (terzo indirizzo della WZ, 0100 in codifica one-hot).

La sua codifica sarà quindi WZ_BIT & WZ_NUM & WZ_OFFSET con:

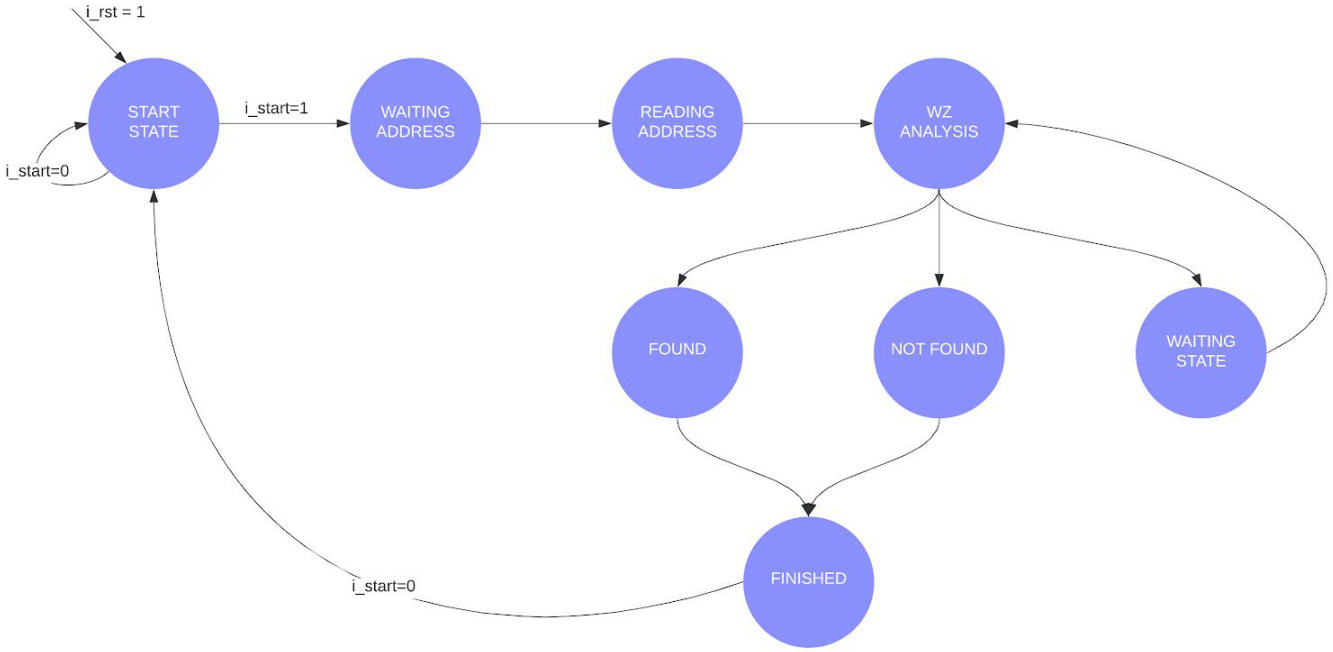
- WZ_BIT = 1
- WZ_NUM = 100
- WZ_OFFSET = 0100

Il risultato è quindi *11000100* in binario, che in decimale corrisponde a 196.

Architettura

Macchina a Stati

Schema

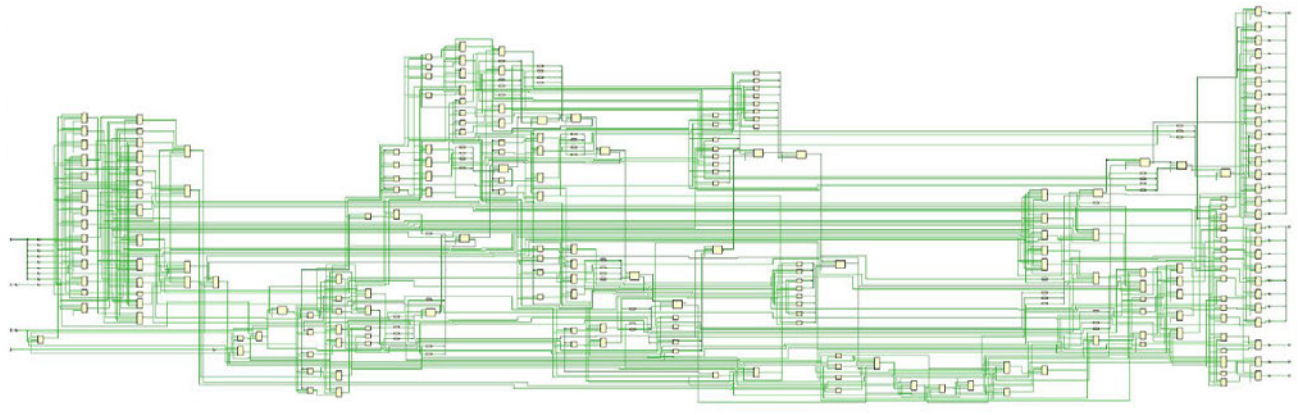


Descrizione Stati

- START_STATE:
    - Stato iniziale che si raggiunge ponendo *i_rst* = 1 e al termine di un'esecuzione completa del programma.
    - Quando *i_start* = 1, si richiede l'indirizzo da codificare alla memoria RAM.
- WAITING_ADDRESS:
    - In attesa dell'indirizzo da codificare da parte della memoria, si richiede l'indirizzo della prima WZ alla memoria RAM.
- READING_ADDRESS:
    - Si memorizza l'indirizzo da codificare in un signal e viene disattivata la lettura da memoria

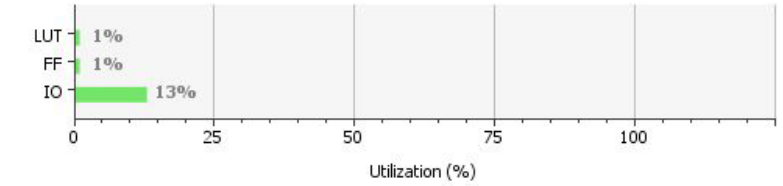
- WZ_ANALYSIS:
    - È lo stato contenente la maggior parte della logica del progetto: *i_data* contiene l'indirizzo di una WZ e durante l'esecuzione si confrontano tutti gli indirizzi appartenenti alla data Working Zone con l'indirizzo da codificare.
    - Nel caso in cui uno degli indirizzi confrontati sia uguale all'indirizzo da codificare si memorizza l'offset rispetto l'indirizzo base della WZ e si passa allo stato *FOUND*.
    - Se nessuno degli indirizzi inclusi nella WZ corrisponde all'indirizzo da codificare e si hanno altre WZ da analizzare allora il contatore viene incrementato e si richiede alla RAM l'indirizzo della WZ successiva. Si passa quindi allo stato *WAITING_STATE*.
    - Se invece nessuno degli indirizzi inclusi nella WZ corrisponde all'indirizzo da codificare ma non ho altre WZ da analizzare (*wz_num* > 6 a causa del ritardo nell'incrementare il *signal*) si passa allo stato *NOT_FOUND*.
- FOUND
    - In questo stato viene attivata la scrittura in memoria mettendo *o_en* = 1 e *o_we* = 1 e l'indirizzo codificato viene posto nella decima cella della RAM. L'indirizzo si ottiene concatenando *1* & *WZ_NUM* & *WZ_OFFSET*. In seguito si pone *o_done* = 1 e si passa allo stato *FINISHED*.
- NOT FOUND
    - Viene attivata la scrittura in memoria e, nella decima posizione della RAM, viene scritto *0* & *ADDR*. Viene settato *o_done* = 1 e si passa allo *FINISHED*.
    - Nell'effettiva implementazione, essendo *ADDR* un indirizzo a 7 bit memorizzato usando 8 bit, il bit più significativo è sempre posto a *0*: non è quindi necessario concatenare *0* e *ADDR*.
- WAITING STATE
    - Si disattiva la lettura dalla RAM mentre si attende l'indirizzo della WZ successiva. Al termine, si ritorna allo stato *WZ_ANALYSIS.*
- FINISHED
    - Il componente ha concluso la codifica dell'indirizzo e quando *i_start* sarà messo a 0, verrà posto anche *o_done* = 0, reinizializzato wz_num e si ritornerà infine allo stato *START_STATE*.

Risultati sperimentali



Il componente HW sviluppato è risultato sintetizzabile ed implementabile. Il suo schema (Post-Sintesi) si può osservare nella figura appena sopra.
In totale vengono utilizzate 154 Lookup Tables, 74 Flip Flop e 38 porte I/O.
Il componente supera i test bench effettuati sia in pre-sintesi sia post-sintesi funzionale.

Resource	Utilization	Available	Utilization %
LUT	154	134600	0.11
FF	74	269200	0.03
IO	38	285	13.33



Simulazioni

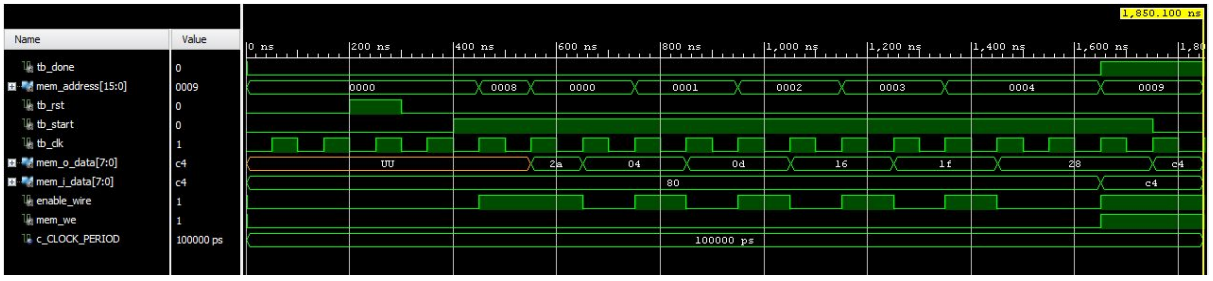
Per verificare il corretto funzionamento del componente sviluppato abbiamo scritto ed eseguito diverse test bench, sia in Behavioural sia in Post-synthesis.
Riportiamo di seguito i più significativi.
Tutti le immagini e i grafici riportati fanno riferimento all'esecuzione in post-synthesis functional.

Test base

- #1: ADDR appartiene a una WZ

  Semplice test di verifica in situazioni standard, in cui viene trovata una WZ che contiene l'indirizzo da codificare.

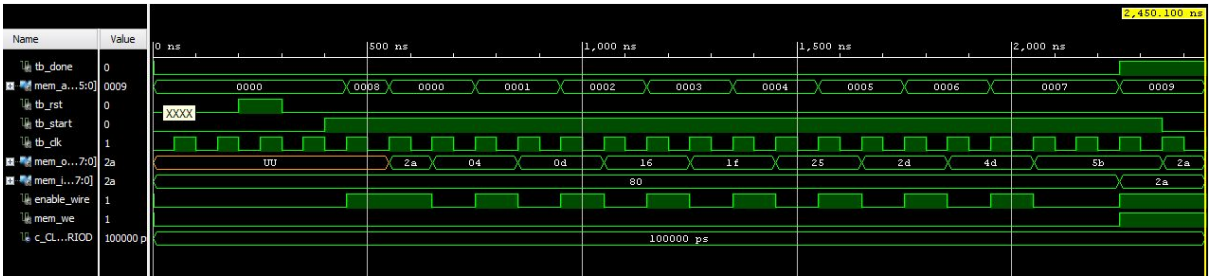
  Indirizzo codificato:
  196
  Soluzione:
  196
  Benchmark:
  Behavioural: 1750 ns
  Post-synthesis functional: 1850100 ps
  Post-synthesis timing: 1853737 ps



- #2: ADDR non appartiene a nessuna WZ

  Semplice test di verifica in situazioni standard, in cui non viene trovata una WZ contenente l'indirizzo da codificare.

  Indirizzo codificato:
  42
  Soluzione:
  42
  Benchmark:
  Behavioural: 2350 ns
  Post-synthesis functional: 2450100 ps
  Post-synthesis timing: 2453737 ps



Test specifici

- #3: Esecuzioni consecutive

  In questo test vengono effettuate due esecuzioni consecutive per verificare che non ci siano conflitti tra le due esecuzioni.

  Indirizzi codificati:

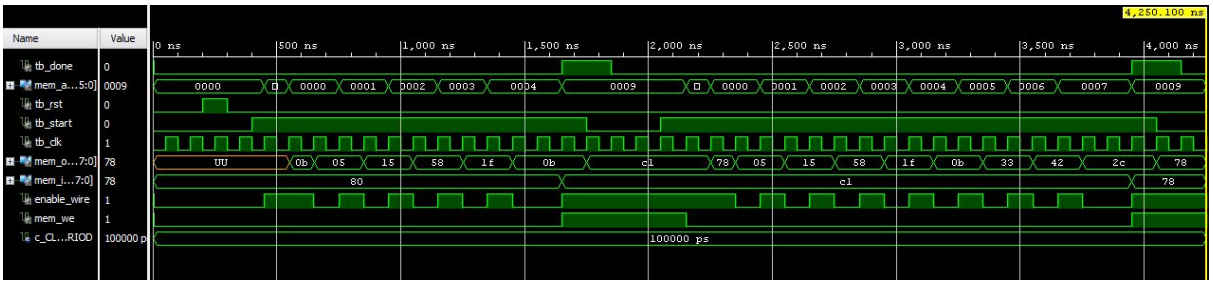
  193
  120

  Soluzioni:

  193
  120

  Benchmark:

  Behavioural: 3950 ns
  Post-synthesis functional: 4250100 ps
  Post-synthesis timing: 4253737 ps



- #4: Reset asincrono

  In questo test, viene verificato il corretto funzionamento nel caso in cui il segnale di reset venga alzato in un qualunque momento/stato dell'esecuzione.

  Indirizzo codificato:

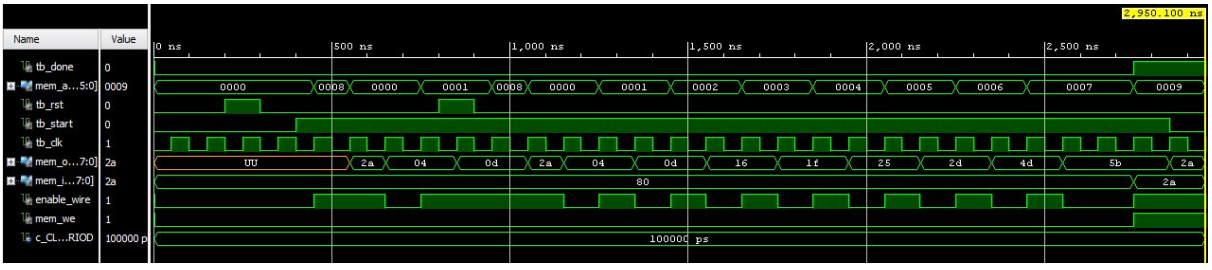
  42

  Soluzione:

  42

  Benchmark:

  Behavioural: 2850 ns
  Post-synthesis functional: 2950100 ps
  Post-synthesis timing: 2953737 ps



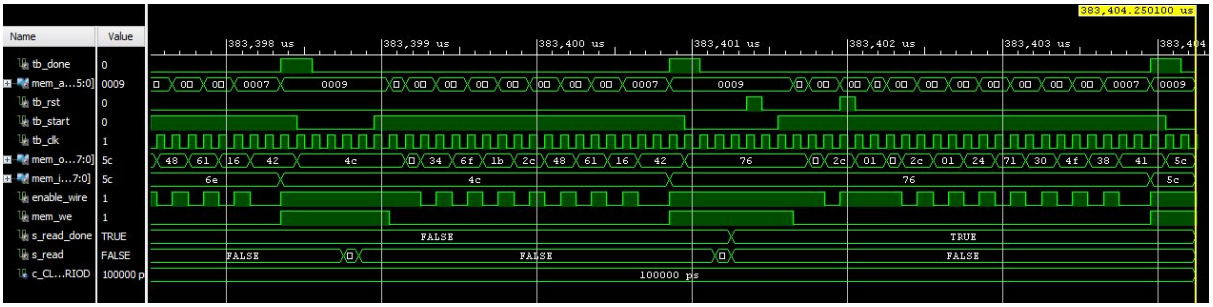
- #5: Stress test

  È stata eseguita una serie di test volta a mettere sotto stress il componente: dopo aver scritto una test bench in grado di leggere i valori della RAM da file, sono stati generati file con un alto numero di test case da eseguire in serie.

  Esempio di un file da 150’000 test con reset asincroni:

  Benchmark:

  Behavioural: 353404350 ns
  Post-synthesis functional: 383404250100 ps
  Post-synthesis timing: 383404253737 ps



Conclusioni

In seguito ad un'accurata e attenta analisi del problema è stata sviluppata una macchina a stati in grado di assolvere al compito richiesto. Successivamente, durante l'implementazione, la macchina astratta è stata leggermente modificata per comunicare correttamente con l'interfaccia fornita dalle specifiche (aggiungendo stati di attesa per la lettura da RAM).

Il componente è stato implementato con un singolo process per una maggiore chiarezza e un minore utilizzo di signal.

Il design è risultato sintetizzabile ed implementabile, abbiamo però notato un "significativo" numero di Lookup Tables (probabilmente dovuto anche all'utilizzo di un singolo process) ma cioè non è sembrato allarmante visto il numero complessivo di LUT implementabile sulla scheda.

Inoltre il componente ha superato con successo ogni test a cui è stato sottoposto dimostrando affidabilità e stabilità.