Progetto di Reti Logiche - Anno 2019/2020
Prof. Gianluca Palermo

Alessandro Polidori (Codice persona 10573078, Matricola 891817)
Olimpia Rivera (Codice persona 10617517, Matricola 892881)

Indice

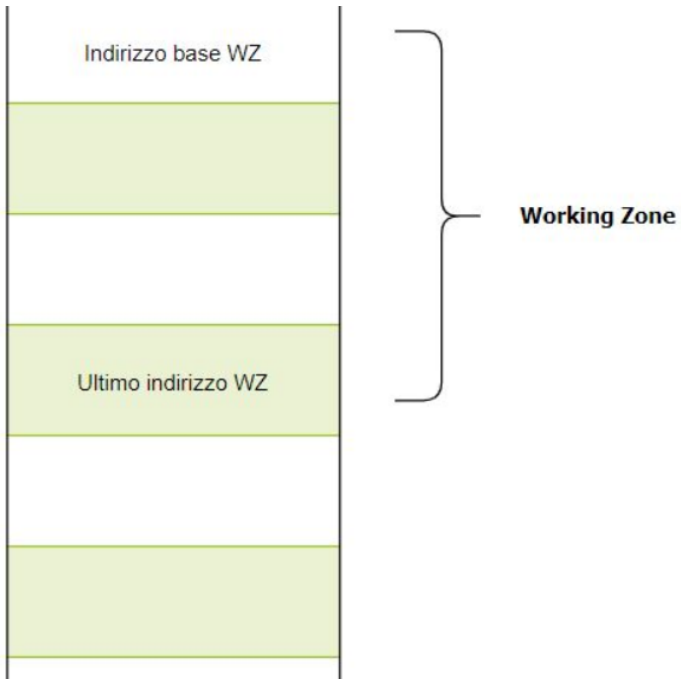
1 Introduzione

1.1 Obiettivo

Viene richiesto di descrivere un componente hardware in linguaggio vhdl sintetizzabile in grado di leggere e scrivere da una RAM già fornita ed eventualmente di modificare indirizzi con il metodo di codifica a bassa dissipazione "Working Zone".

1.2 Working Zone

Nel caso esaminato le working-zone sono otto intervalli di memoria di dimensione fissa (4 indirizzi compreso quello base).



1.3 Codifica indirizzo

L'indirizzo letto dalla RAM dovrà essere modificato nel caso in cui sia contenuto in una delle working-zone. L'indirizzo modificato è dato dalla concatenazione di un '1', del numero di working-zone (su 3 bit) e dell'offset rispetto all'indirizzo base della wz in codifica one-hot. Se l'indirizzo non è contenuto in alcuna working-zone, allora viene trasmesso così come è.

1.4 Specifica del progetto

Il componente da descrivere deve avere la seguente interfaccia:

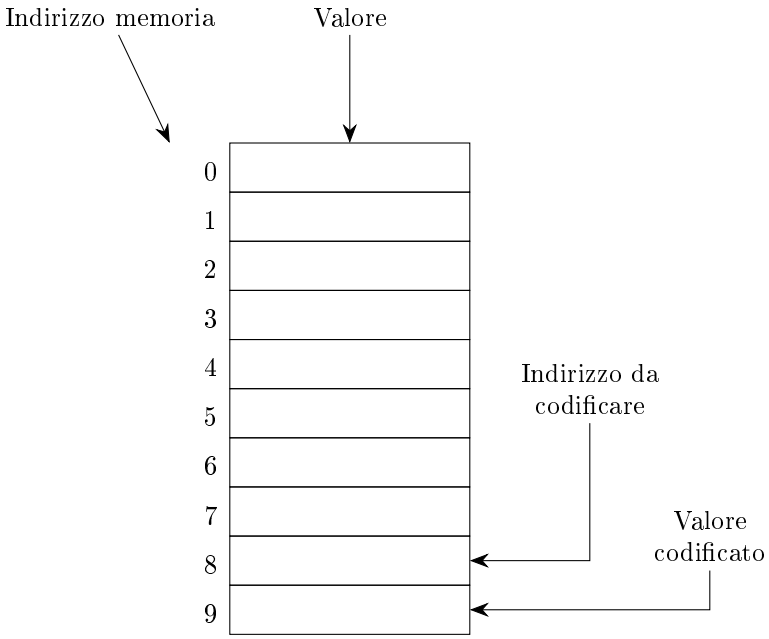
```
entity project_reti_logiche is
      port (
            i_clk             : in  std_logic;
            i_start           : in  std_logic;
            i_rst             : in  std_logic;
            i_data            : in  std_logic_vector(7 downto 0);
            o_address         : out std_logic_vector(15 downto 0);
            o_done            : out std_logic;
            o_en              : out std_logic;
            o_we              : out std_logic;
            o_data            : out std_logic_vector (7 downto 0)
      );
end project_reti_logiche;
```

Dove:

- i_clk è il segnale di CLOCK generato dal test-bench
- i_start è il segnale di START generato dal test-bench
- i_rst è il segnale di RESET che inizializza la macchina
- i_data è il segnale che arriva dalla memoria in seguito ad una richiesta di lettura
- o_address è il segnale di uscita che manda l'indirizzo alla memoria
- o_done è il segnale di uscita che comunica la fine dell'elaborazione e il dato di uscita scritto in memoria
- o_en è il segnale di ENABLE da dover mandare alla memoria per poter comunicare
- o_we è il segnale di WRITE ENABLE da dover mandare alla memoria (='1') per poterci scrivere. Per leggere, questo segnale deve essere posto a '0'
- o_data è il segnale di uscita dal componente verso la memoria

1.5 Memoria

Il componente andrà a lavorare solo sui primi dieci indirizzi di memoria RAM. I primi 8 contengono gli indirizzi base delle working-zone, il nono l'indirizzo da modificare. In corrispondenza del decimo indirizzo, invece, si troverà il valore scritto in seguito alla codifica. I valori delle WZ possono cambiare in seguito ad un segnale di RESET.
La memoria è già istanziata all'interno dei testbench forniti ed è derivata dalla User guide di VIVADO disponibile al seguente link : https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_3/ug901-vivado-synthesis.pdf



2 Architettura

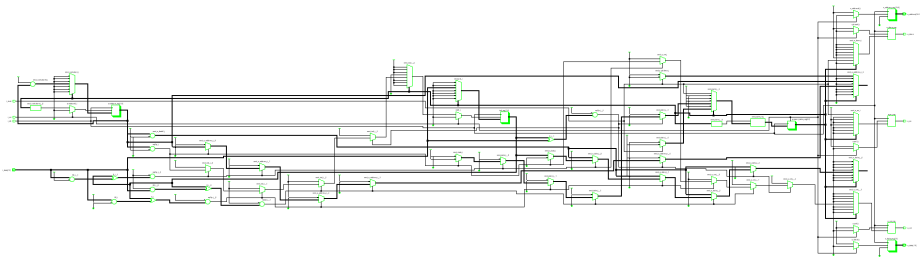


figure: schematic RTL di Vivado

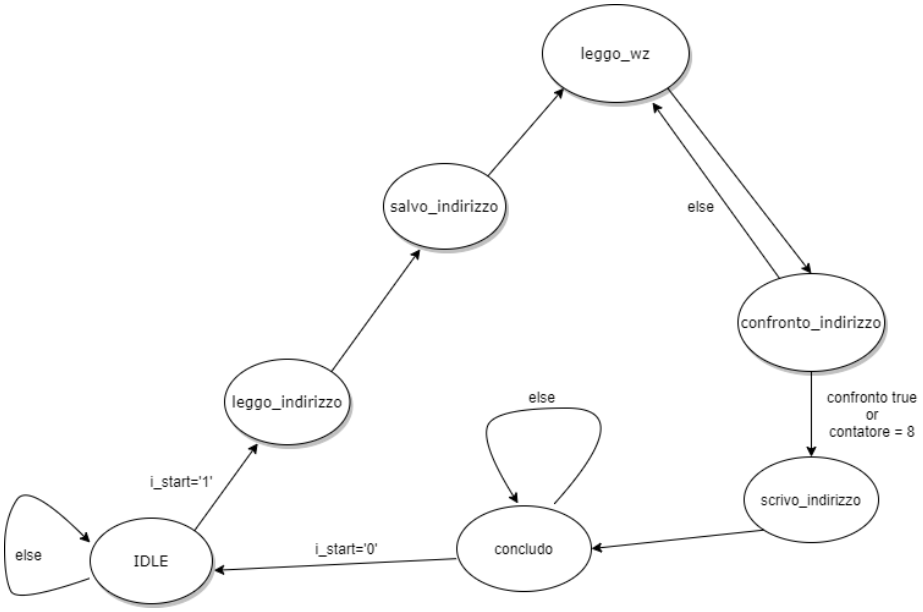
2.1 Implementazione

Abbiamo deciso di implementare una macchina a stati attraverso due processi, uno per gestire la parte sequenziale (denominato "registri") ed uno per la descrizione degli stati e la scelta dello stato prossimo (denominato "transizioni"). Inizialmente erano presenti delle variabili all'interno del processo "transizioni", ma al fine di dare una struttura più coerente al codice sono state tutte sostituite da segnali e segnali prossimi corrispondenti. L'implementazione finale obbliga il componente a rileggere dopo ogni START gli indirizzi di base delle working-zone. Questa scelta penalizza i tempi di calcolo, ma ci ha permesso di ottimizzare l'uso della memoria (è necessario mantenere un solo indirizzo).

2.2 Macchina a stati

- IDLE è lo stato iniziale. La codifica inizia quando il segnale START viene portato a '1'. Viene dato per scontato che, da qualunque stato, se RESET viene portato a '1' la macchina torna in IDLE
- leggo_indirizzo: viene letto dalla ram l'indirizzo da modificare
- salvo_indirizzo: l'indirizzo letto viene salvato in un registro
- leggo_wz: viene letto l'indirizzo base delle wz

- confronto_indirizzo serve a confrontare l'indirizzo con i vari indirizzi delle working-zone. Si passa allo stato "scrivo_indirizzo" se il confronto ha esito positivo o se l'indirizzo è stato confrontato con tutti gli indirizzi della working-zone.
- scrivo_indirizzo: si scrive l'indirizzo nella memoria ram
- concludo è lo stato in cui DONE è a '1' e si aspetta che il segnale START venga portato a '0' per poter tornare ad IDLE.



2.3 Ottimizzazioni

Nella prima prima versione della macchina erano presenti anche due stati chiamati "cambio_indirizzo" (nel quale veniva modificato o, eventualmente, mantenuto uguale l'indirizzo) e "salvo_wz". Ci siamo però resi conto che era possibile integrare tutte le funzioni di "cambio_indirizzo" all'interno di "confronto_indirizzo" e che era possibile bypassare completamente lo stato "salvo_wz". Questo ci ha permesso di guadagnare numerosi cicli di clock (uno per codifica per "cambio_indirizzo" e otto per codifica nel worst-case per "salvo_wz").

3 Risultati della sintesi

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed
synth_1	constrs_1	synth_design Complete!								52	30	0.00	0	0	3/20/20, 10:55 PM	00:00:13
impl_1	constrs_1	Not started														

```
INFO: [Common 17-83] Releasing license: Synthesis
22 Infos, 19 Warnings, 0 Critical Warnings and 0 Errors encountered.
synth_design completed successfully
synth_design: Time (s): cpu = 00:00:12 ; elapsed = 00:00:13 . Memory (MB): peak = 781.656 ; gain = 462.160
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 781.656 ; gain = 0.000
```

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	52	0	134600	0.04
LUT as Logic	52	0	134600	0.04
LUT as Memory	0	0	46200	0.00
Slice Registers	33	0	269200	0.01
Register as Flip Flop	30	0	269200	0.01
Register as Latch	3	0	269200	<0.01
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

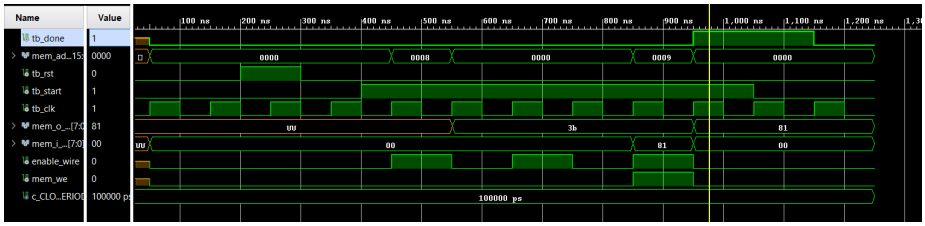
4 Casi di test

Dopo aver superato i test forniti abbiamo voluto mettere alla prova il componente simulando le seguenti situazioni :

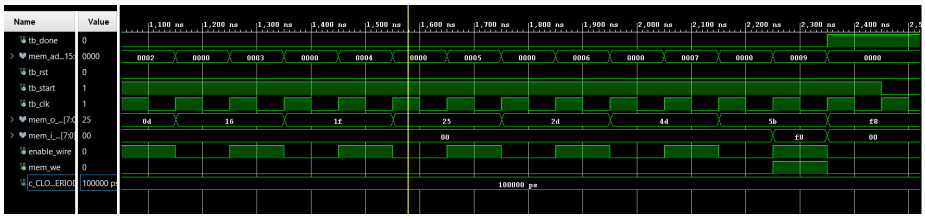
Posizioni indirizzo

- L'indirizzo è presente nel primo "spazio" della prima working-zone
- L'indirizzo è presente nell'ultimo "spazio" della prima working-zone
- L'indirizzo è presente nel primo "spazio" dell'ultima working-zone
- L'indirizzo è presente nell'ultimo "spazio" dell'ultima working-zone
- L'indirizzo è presente in "spazi" casuali di working-zone casuali

primo spazio della prima working-zone



ultimo spazio della ultima working-zone

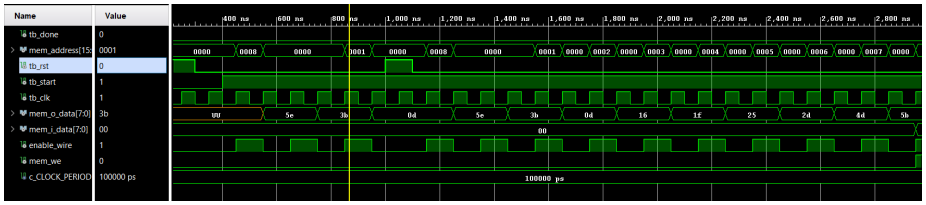


Altre casistiche

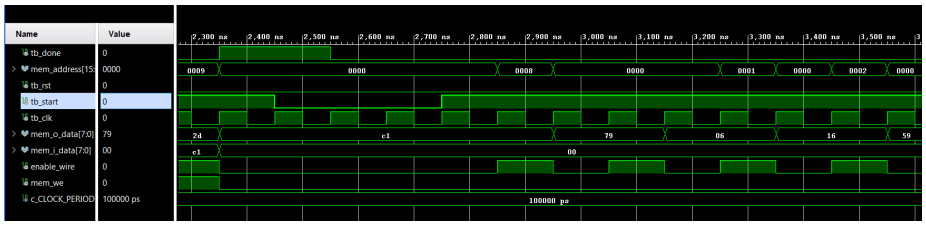
- Nuova codifica. Il componente deve essere in grado di ricevere un secondo indirizzo. Avendo scelto di far rileggere ogni volta i valori contenuti nella ram, la criticità di questo test risiede solo nella fase di "conclusione" della codifica e ritorno in IDLE

- Il segnale di RESET viene portato a '1' durante diversi momenti della codifica. Il componente deve essere in grado di tornare in IDLE in un qualunque momento
- Il segnale START viene portato a '0' (a fine codifica) con un ritardo di alcuni cicli di clock. Abbiamo voluto testare anche la capacità del componente di gestire eventuali ritardi dell'abbassamento di START

Reset durante la codifica



seconda codifica dopo nuovo start



5 Conclusioni

Il componente è in grado di superare i test forniti e quelli sviluppati da noi in modalità Behavioral, Post-Syntesis Functional e Post-Syntesis Timing. Riteniamo quindi di aver raggiunto gli obiettivi prefissati.