POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Relazione Prova finale di Reti Logiche

Luca Pozzato - 10794909, Andrea Rossi - 10799170

Academic year:
2023-2024

Abstract: Lo scopo del progetto è la descrizione in VHDL di un componente hardware che, interfacciandosi con una memoria in cui sono presenti alcuni dati, esegue un'analisi su essi e completa i dati mancanti della memoria stessa.

1. Introduzione

1.1. Specifiche

Il componente riceve in ingresso una sequenza di `K` parole `W` il cui valore è compreso tra 0 e 255, dove il valore zero va interpretato come *"valore non specificato"*. Le parole sono memorizzate a partire da un indirizzo `ADD` ogni due byte (e.g. `ADD, ADD+2... ADD+2*(K-1)`). Il byte mancante (agli indirizzi `ADD+1, ADD+3..., ADD+2(K-1)+1`) dovrà essere completato con il valore di *credibilità* `C` della relativa parola `W`: essa può assumere valori tra 31 e 0, è pari a 31 ogni volta che il valore `W` è *valido* (diverso da zero) mentre viene decrementato rispetto al valore precedente nel caso in cui si incontri uno zero in `W`.
L'obiettivo del componente è processare la stringa e modificarla come segue:

- sostituire le parole con valore 0 con l'ultimo valore *valido*, cioè l'ultimo valore della sequenza letto diverso da zero;
- completare per ogni parola il byte mancante con il suo valore di *credibilità* `C`.

Un esempio di sequenza in ingresso è ad esempio (con valori in decimale):

120	0	**10**	0	**0**	0	**0**	0	**16**	0

Stringa in ingresso con `K = 5` parole (in **grassetto**)

120	*31*	**10**	*31*	***10***	*30*	***10***	*29*	**16**	*31*

Stringa processata (in *corsivo* le modifiche)

Da completare con un esempio in grande

2. Architettura

2.1. Interfaccia e funzionamento

Il modulo ha la seguente interfaccia:

- tre segnali principali
  - `i_start`: 1 bit, segnale di inizio elaborazione;
  - `i_add`: 16 bit, indirizzo della memoria da cui inizia la porzione da processare;
  - `i_k`: 10 bit, numero di parole da elaborare;
- due ingressi ausiliari
  - `i_clk`: 1 bit, segnale di clock unico per tutto il sistema;
  - `i_rst`: 1 bit, segnale di reset asincrono;
- un'uscita principale
  - `o_done`: 1 bit, segnale di terminata elaborazione;
- cinque connessioni con la memoria
  - `i_mem_data`: 8 bit, contenuto in ingresso dalla memoria quando letta;
  - `o_mem_en`: 1 bit, segnale di *enable*, deve essere 1 per comunicare con la memoria;
  - `o_mem_we`: 1 bit, segnale di *write enable*, deve essere 1 per scrivere, 0 per leggere;
  - `o_mem_addr`: 16 bit, indirizzo della memoria da cui leggere o in cui scrivere;
  - `o_mem_data`: 8 bit, dato che verrà scritto in memoria;

Tutti i segnali sono sincroni e devono essere interpretati sul fronte di salita del clock; fa eccezione il segnale di reset che è asincrono.

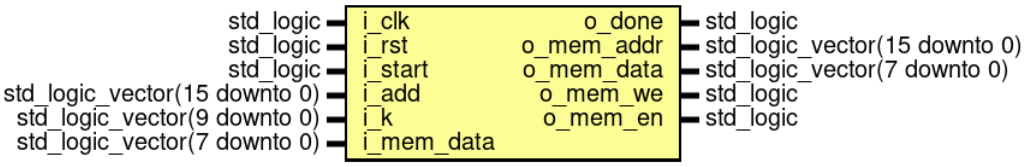

Diagramma del componente

3. Risultati sperimentali

3.1. Report di sintesi

Dopo aver sintetizzato il design del progetto, il circuito è stato simulato utilizzando il test bench fornito. Il circuito si comporta correttamente, superando il test sia con la simulazione comportamentale (behavioral simulation) che con la simulazione post-sintesi (post-synthesis simulation). Di seguito, sono riportate le tabelle con il timing report e l'utilizzo delle risorse. Nella Tabella 1 è mostrato il timing report del design; analizzandolo, si può notare che il Worst Negative Slack (WNS) è di 15,413 ns. Un WNS di 15,413 ns con un periodo di clock di 20 ns indica che il circuito impiega (20 ns - 15,413 ns = 4,587 ns) per completare l'esecuzione di uno stato della FSM del design, dato che la FSM è stata sincronizzata in modo da eseguire uno stato per ogni ciclo di clock, come è possibile notare dalla Figura 1.

Setup	**Hold**	**Pulse Width**
Worst Negative Slack (WNS): 15,413 ns	Worst Hold Slack (WHS): 0,148 ns	Worst Pulse Width Slack (WPWS): 9,500 ns
Total Negative Slack (TNS): 0,000 ns	Total Hold Slack (THS): 0,000 ns	Total Pulse Width Negative Slack (TPWS): 0,000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 119	Total Number of Endpoints: 119	Total Number of Endpoints: 79

Tabella 1: Design Timing Summary

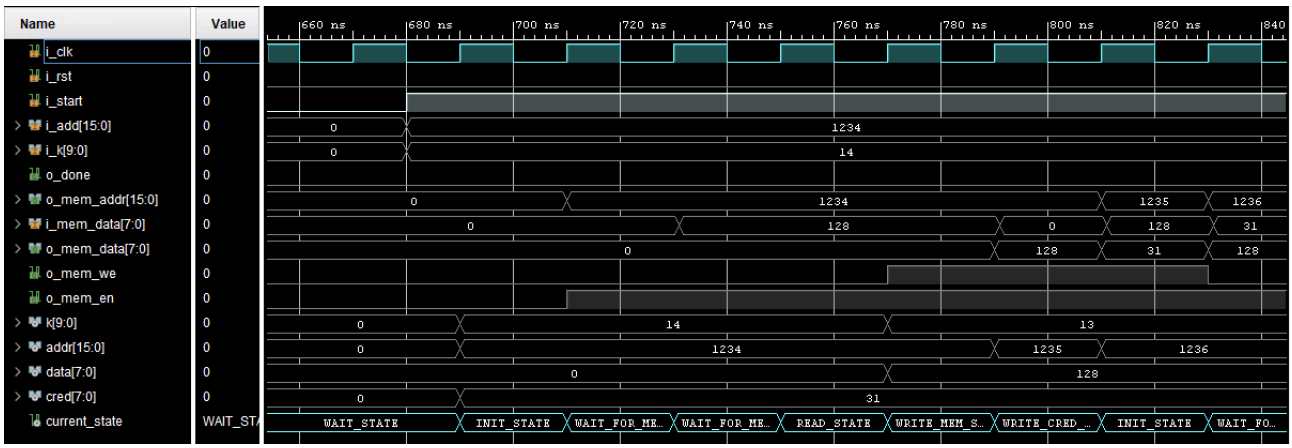

Figura 1: sincronizzazione della FSM con il clock

Da questo valore è anche possibile calcolare il periodo di clock minimo che il design può supportare. Sapendo che la memoria (RAM) ha un ritardo massimo di 2 ns nelle operazioni di scrittura e lettura, il periodo di clock minimo è dato da:

$$T_{clk} = 20ns + 2ns - 15,413ns = 6,587ns \tag{1}$$

e la frequenza di clock massima supportata dal design è:

$$f_{clk} = \frac{1}{T_{clk}} = \frac{1}{6,587ns} \approx 151,8MHz \tag{2}$$

Dalla Tabella 2 si può invece osservare come il design utilizzi 78 registri di tipo Flip-Flop e 86 LUTs, il che significa che il design non presenta alcun tipo di Latch. Questo evita comportamenti imprevedibili e indesiderati, assicurando inoltre che il design sia completamente sincrono. Per evitare la creazione di Latch e sincronizzare correttamente il design con il periodo di clock, è stato necessario implementare la FSM in un singolo processo invece di dividerla in più processi.

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	78	0	269200	0,03
Register as Flip Flop	78	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

Tabella 2: Resource Utilization Summary

3.2. Simulazioni

Per assicurarsi che il design fosse corretto e privo di errori, sono stati simulati vari test bench con l'obiettivo di verificare diversi corner cases. Inoltre, è stato sviluppato un codice in Python per generare automaticamente una serie di test bench, inizializzando la RAM con messaggi composti da $K \in [1, 100]$ parole e assegnando a ciascuna parola un valore $W \in [0, 255]$, come da specifica. I test bench generati da Python sono stati successivamente simulati con GHDL, uno strumento utilizzato per la simulazione e verifica dei design VHDL.

3.2.1 Tesh bench fornito

In questo test bench viene illustrato il funzionamento del design con una sequenza definita come 'normale', in cui viene mostrato il corretto utilizzo di tutti i segnali per interfacciarsi con la RAM. Nella Figura 2 e nella Figura 3 è possibile osservare i segnali di inizio e fine elaborazione, che vengono attivati correttamente durante l'esecuzione del design. In particolare, il segnale di inizio elaborazione (i_start), attivato una volta che il segnale di reset (i_rst) è disattivato, avvia correttamente l'elaborazione del design, come si può notare dal cambiamento degli stati; in dettaglio, la FSM passa da WAIT_STATE a INIT_STATE, procedendo con la lettura della RAM. Una volta che il design ha terminato l'elaborazione della sequenza, il segnale di terminazione dell'elaborazione (o_done) viene attivato e, infine, il segnale i_start viene disattivato, rendendo il design pronto per una nuova elaborazione.

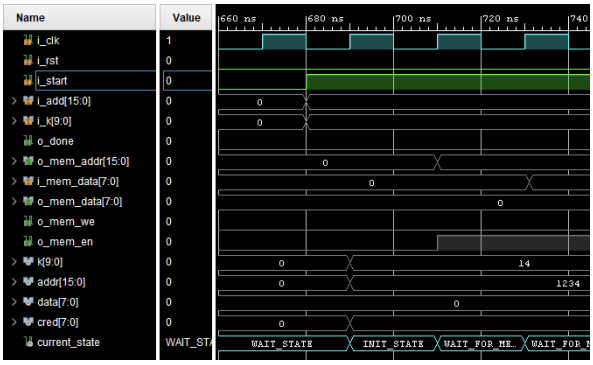

Figura 2: Segnali di inizio elaborazione

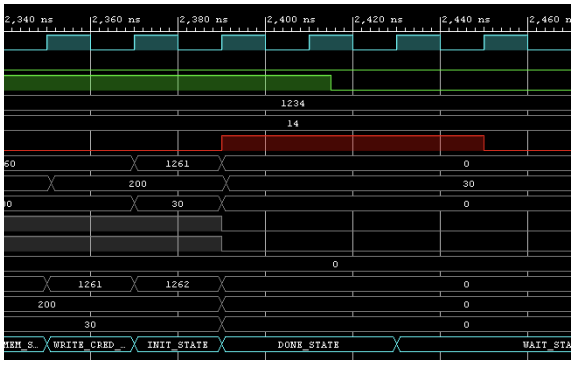

Figura 3: Segnali di fine elaborazione

3.2.2 Sequenza di lunghezza nulla

In questo test bench viene illustrato il comportamento del design quando la sequenza è vuota, ovvero quando il segnale che indica la lunghezza della sequenza (i_k) è pari a 0. Come mostrato nella Figura 4, il design avvia correttamente l'elaborazione non appena il segnale di reset viene disattivato e quello di start attivato. Tuttavia, appena rileva che k = 0 leggendo dalla RAM, il design conclude immediatamente l'elaborazione, attivando il segnale o_done, che viene poi disattivato quando i_start è disattivato. La FSM, una volta attivato i_start, passa da WAIT_STATE a INIT_STATE; da quest'ultimo stato, procede a DONE_STATE se k = 0, altrimenti passa a WAIT_FOR_MEM_STATE, come si può osservare nella figura.

3.2.3 Credibilità che raggiunge 0

Questo test bench illustra il comportamento del design quando la credibilità viene decrementata ogni volta che si incontra uno zero in W, finché non si rileva un valore diverso da 0, momento in cui la credibilità viene reinizializzata a 31 per poi essere nuovamente decrementata se si incontra un altro zero. Secondo la specifica, una volta che la credibilità raggiunge 0, il design deve mantenere tale valore fino a che non si rileva un valore in W diverso da 0 o fino al termine della sequenza. Come previsto, alla fine della sequenza, il design deve attivare il segnale di fine elaborazione. Nella Figura 5 e nella Figura 6 è possibile osservare come il design si comporti correttamente, continuando a decrementare la credibilità fino a 0 e mantenendola a 0, dato che non si verificano valori diversi da 0 in W.

3.2.4 Test bench con sequenza nulla iniziale

In questo test bench si è voluto verificare il funzionamento del design quando la sequenza di parole W inizia con un valore pari a 0. In questo scenario, il design deve avviare l'elaborazione della sequenza mantenendo il

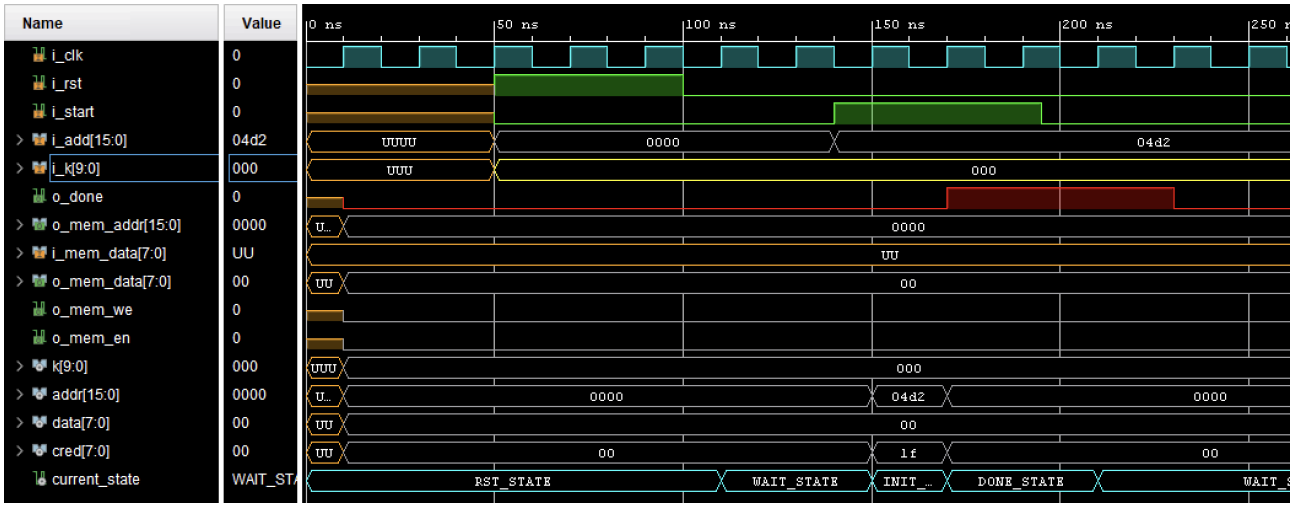

Figura 4: Sequenza di lunghezza nulla

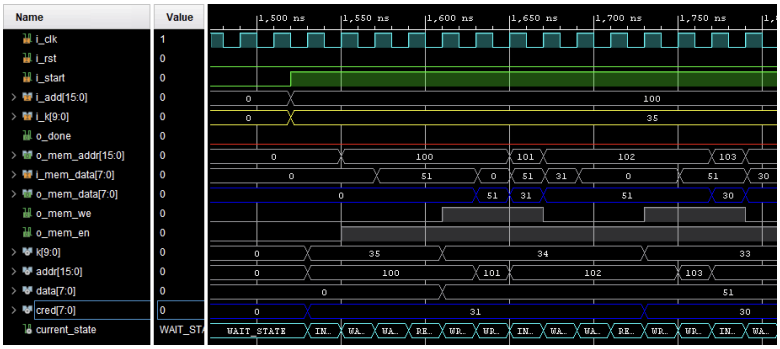

Figura 5: Segnali di inizio elaborazione

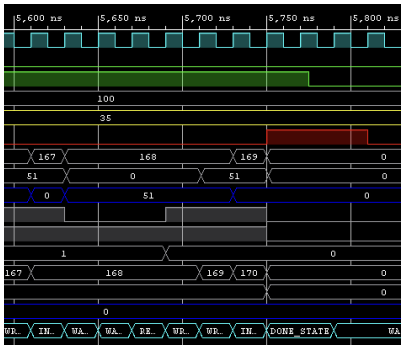

Figura 6: Segnali di fine elaborazione

valore W nullo e impostando e mantenendo la credibilità a 0, fino a quando non si legge dalla RAM un valore di W diverso da 0. A quel punto, il design si comporta come mostrato nei test bench precedenti. Nella Figura 7 e nella Figura 8 si verifica che il design opera in modo appropriato: in particolare, si può notare come sia la credibilità che il valore di W rimangano a 0. Questo è evidenziato anche dal segnale o_mem_data (segnale che indica il valore da scrivere nella RAM all'indirizzo o_mem_addr), che resta a 0 fino a quando non viene letto dalla RAM un valore diverso da 0; nel caso del test bench, questo valore è 2 e la credibilità viene poi posta a 31.

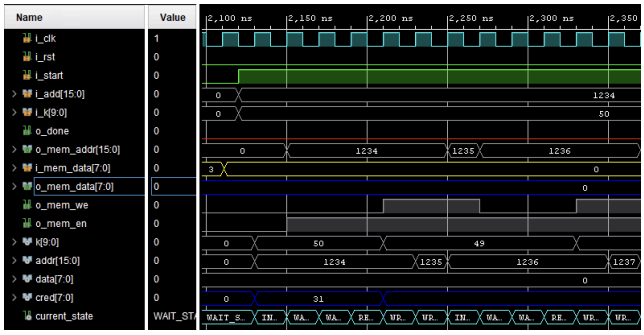

Figura 7: Segnali di inizio elaborazione

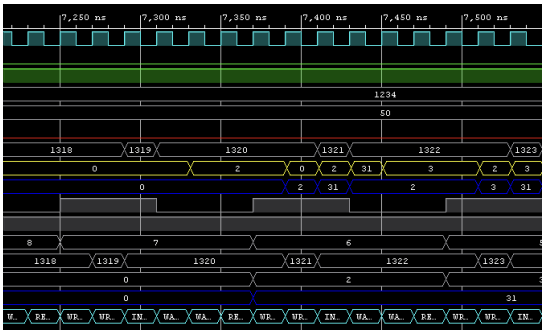

Figura 8: Fine elaborazione di una sequenza di 0

3.2.5 Sequenze multiple

In questo test bench si verifica come il design si comporta nell'elaborazione di sequenze multiple. Dalla specifica, il design, una volta completata l'elaborazione della prima sequenza, deve attivare il segnale di fine elaborazione (o_done). Quando o_done viene attivato, il segnale i_start viene disattivato, seguito dalla disattivazione del segnale di fine elaborazione, il che indica che il modulo è pronto per una nuova elaborazione. Quando i_start viene attivato una seconda volta, il modulo avvia la seconda elaborazione. Secondo la specifica, la seconda elaborazione non deve attendere il reset del modulo; questo è verificabile osservando come il modulo, una volta terminata l'elaborazione, passi da DONE_STATE a WAIT_STATE, e successivamente, con la riattivazione di i_start, la FSM passi da WAIT_STATE a INIT_STATE, procedendo con l'elaborazione della nuova sequenza.

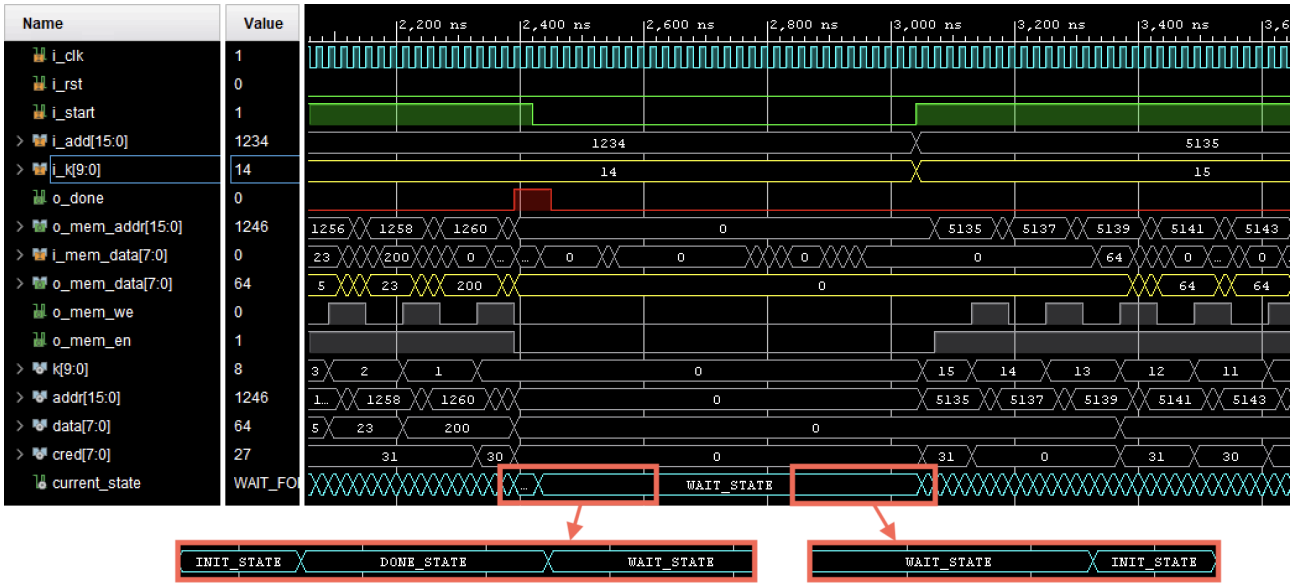

Figura 9: Sequenze multiple

3.2.6 Segnale di reset

In questo test bench si verifica il corretto funzionamento del design a fronte del segnale di reset asincrono.

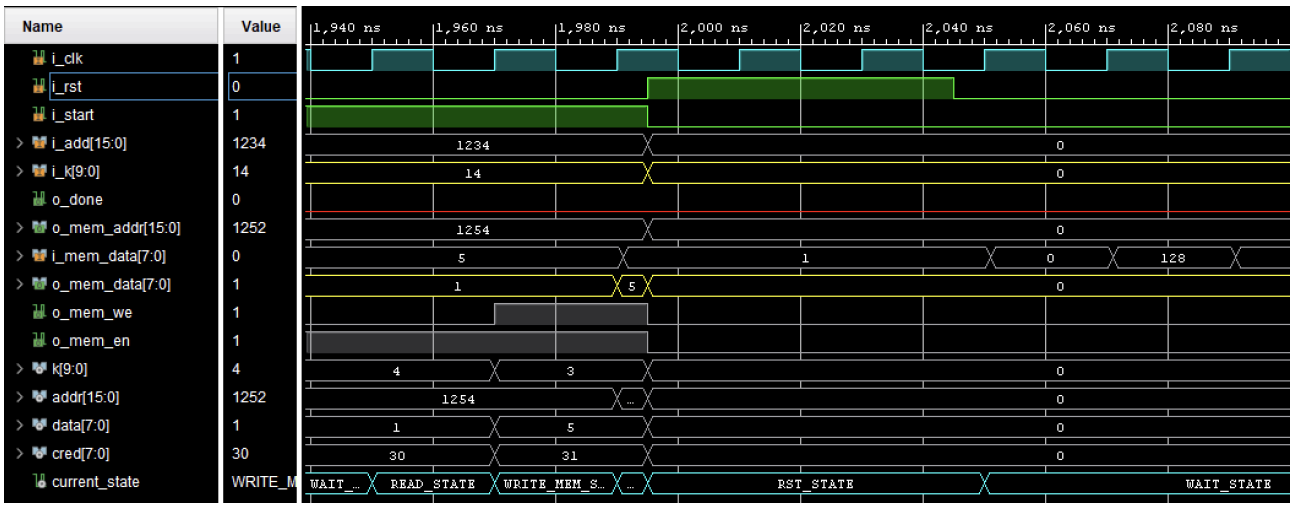

Figura 10: Segnale di reset